THÈSE DE DOCTORAT DE
SORBONNE UNIVERSITÉ
préparée à EURECOM

École doctorale EDITE de Paris n° ED130
Spécialité: «Informatique, Télécommunications et Électronique»

Sujet de la thèse:

The Phantom Thread Between Malleable and Non-Malleable Cryptography

Thèse présentée et soutenue à Biot, le 31/03/2025, par

Luigi Russo

Président	Prof. Duong Hieu Phan	Telecom Paris, IP Paris
Rapporteurs	Prof. Carla Ràfols	Universitat Pompeu Fabra
	Prof. Dario Catalano	University of Catania
Directeurs de thèse	Prof. Melek Önen	EURECOM
	Prof. Antonio Faonio	EURECOM

À l'*Insaisissable*

Abstract

At the forefront of cryptographic techniques that enable secure digital applications are public-key encryption (PKE) schemes and zero-knowledge (ZK) proofs. A PKE allows two parties without a shared secret to communicate securely, while ZK proofs validate statements without revealing anything beyond their validity.

Malleability, in cryptographic terms, refers to the possibility of efficiently modifying and transforming, in a predictable way, an encrypted message or a proof; it is a desirable feature that finds applications in many settings, such as privacy-preserving outsourced storage and computation. On the other hand, non-malleability is defined as the property that prevents such predictable modifications that, in a general-purpose cryptosystem, may lead to vulnerabilities and undesired attacks.

This thesis aims to explore the nuanced interplay and the relationship between malleability and non-malleability. With a combination of theoretical analysis and practical case studies, it provides insights into the boundaries and connections between these two properties and how they can coexist in secure and efficient protocols.

In particular, we show that a wide class of non-interactive succinct arguments (zkSNARKs), a special class of ZK proof systems, has strong non-malleable properties even if derived from malleable cryptographic primitives, like homomorphic polynomial commitments: our results apply to popular proof systems, such as Plonk and Marlin, and the KZG commitment scheme.

We provide a framework for analyzing the non-malleability of modular zkSNARKS, with a special emphasis on optimized schemes and flexible architectures, such as those based on the paradigm of Virtual Machines.

Also, we propose efficient and secure protocols based on a class of malleable PKEs, the Re-randomizable RCCA (Rand-RCCA) PKEs, with applications to electronic voting and anonymous e-mail, among others.

Last but not least, we initiate the study of tight security in the Rand-RCCA setting, thus giving insight into how tight the security of these schemes translates to the trust that we have with respect to standard cryptographic assumptions.

Résumé

À la pointe des techniques cryptographiques qui permettent des applications numériques sécurisées se trouvent les schémas de chiffrement à clé publique (PKE) et les preuves à divulgation nulle de connaissance (ZK). Un PKE permet à deux parties sans secret partagé de communiquer en toute sécurité, tandis que les preuves ZK valident des déclarations sans révéler quoi que ce soit au-delà de leur validité.

La malléabilité, en termes cryptographiques, fait référence à la possibilité de modifier et de transformer efficacement, de manière prévisible, un message chiffré ou une preuve ; c'est une caractéristique souhaitable qui trouve des applications dans de nombreux contextes, tels que le stockage et le calcul délégués préservant la vie privée. En revanche, la non-malléabilité est définie comme la propriété qui empêche de telles modifications prévisibles qui, dans un système cryptographique à usage général, peuvent entraîner des vulnérabilités et des attaques indésirables.

Cette thèse vise à explorer l'interaction nuancée et la relation entre la malléabilité et la non-malléabilité. À l'aide d'une combinaison d'analyses théoriques et d'études de cas pratiques, elle fournit des éclaircissements sur les limites et les connexions entre ces deux propriétés et sur la manière dont elles peuvent coexister dans des protocoles sécurisés et efficaces.

En particulier, nous montrons qu'une large classe de systèmes de preuves ZK (zkSNARKs) possède de fortes propriétés de non-malléabilité même si elle est dérivée de primitives cryptographiques malléables, comme les engagements polynomiaux homomorphes : nos résultats s'appliquent à des systèmes de preuves populaires, tels que Plonk et Marlin, ainsi qu'au schéma d'engagement KZG.

Nous proposons un cadre pour analyser la non-malléabilité des zkSNARKS modulaires, avec une attention particulière portée sur les schémas optimisés et les architectures flexibles, telles que celles basées sur le paradigme des machines virtuelles.

De plus, nous proposons des protocoles efficaces et sécurisés basés sur une classe de PKE malléables, les PKE RCCA ré-randomisables (Rand-RCCA), avec des applications à la vote électronique et aux e-mails anonymes, entre autres.

Enfin, nous entamons l'étude de la sécurité stricte dans le cadre Rand-RCCA, offrant ainsi un aperçu de la manière dont la solidité de la sécurité de ces schémas se traduit par la confiance que nous avons vis-à-vis des hypothèses cryptographiques standard.

Contents

Chapter 1

Introduction

In the past few decades, our society has experienced a significant shift towards a digital-focused world. Almost every task, whether it involves sending messages, making purchases, participating in elections, or signing contracts, is now conducted through digital means. As this vast digital landscape presents numerous opportunities for dishonest or malicious activities, ensuring secure transactions becomes crucial. The field of *Cryptography* serves as the foundation for addressing this challenge, by exploring the principles and limitations of digital security.

1.1 Cryptography: An overview

Cryptography has a long and fascinating history that traces back to the early centuries of the human race. In this manuscript, we do not address in detail the role cryptography has played throughout history, but we refer the reader to the famous book *The Codebreakers* written by the historian David Kahn [Kah67] for an in-depth overview.

Classical Cryptography. Etymologically, the word *cryptography* comes from the Ancient Greek κρυπτός ("hidden, secret") and γράφειν ("to write"). Until late in the 20th century, cryptography was effectively a synonym of "encryption", namely the process of converting a message (the "plaintext") to some unintelligible nonsense text (the "ciphertext"), which can only be read by reversing the process (the "decryption").

The early cryptographic schemes were simple enough to be practically computed and solved by hand, and have hence fallen into disuse. We now refer to this period as *Classical Cryptography*, including the naïve systems used since Greek and Roman times, the elaborate Renaissance ciphers, but also the codes developed during World War II, such as the infamous Enigma machine.

Modern Cryptography. Nowadays, cryptography has evolved significantly from its historical roots and encompasses much more than "secret writing": it deals with mechanisms for ensuring data or software integrity, enforcing privacy online, managing digital identities, designing secure protocols for electronic elections, payment systems, decentralized finance (DeFi) applications, and many more. Hence, modern cryptography is the practice and study of techniques for secure communication in the presence of adversarial behavior, and is at the intersection of the disciplines of mathematics, computer science, information security, physics, and many others.

In the following pages, we provide a very brief and high-level overview of some of the main primitives and security notions in cryptography, such as encryption schemes, zero-knowledge

proofs and secure multi-party protocols.

1.1.1 Encryption Schemes

Secret-Key Cryptography. The development of modern cryptosystems started with Feistel's work at IBM in the early 1970s [Fei73] and culminated in the adoption of the Data Encryption Standard (DES) in 1977, that was widely used for encrypting data. DES employs *symmetric-key* cryptography, where the same key is used for both encryption and decryption (hence the name symmetric). Clearly, in this setting the key must remain *secret*, as it is used to encrypt and decrypt data.

However, a significant challenge is the management of keys, as each pair of communicating parties ideally needs a unique key, complicating secure key establishment without a pre-existing secure channel.

Public-Key Cryptography. A pivotal moment in cryptography occurred in 1976 when Whitfield Diffie and Martin E. Hellman introduced the notion of *public-key* cryptography in their seminal paper "New Directions in Cryptography" [DH76]. This innovation addressed the key management issue and was based on the computational hardness of the discrete logarithm problem (see Section 2.3.1).

The main idea of public-key cryptography is to break the symmetry between encryption and decryption keys. In particular, there exists a pair of keys, a public one and a private one. The public one allows anyone to encrypt a message that can only be decrypted with the corresponding private key. The private key, on the other hand, is the only one that must be kept secret.

Although practical implementations of public-key encryption were not available at the time, the idea sparked considerable interest and development in the cryptographic community. In 1978, Rivest, Shamir, and Adleman [RSA78] introduced the first practical public-key encryption scheme. Their scheme, now simply known as RSA, relies on the difficulty of factoring large integers, a problem that challenged mathematicians for centuries. Despite the renewed interest in developing more efficient factoring methods, no significant advancement has compromised the security of the RSA cryptographic system, that is still actively used in practice today. Later on, new constructions were proposed. A notable example is the ElGamal encryption scheme [ElG84] that is based on the hardness of the discrete logarithm problem.

1.1.2 Multi-Party Protocols

While traditional cryptographic tasks primarily focus on ensuring the security and integrity of communication between two users, the security model often portrays the adversary as an entity operating from the *outside* of the system of participants. In this paradigm, indeed, adversaries are typically viewed as eavesdroppers who attempt to intercept and glean information from the exchanges between the sender and receiver. This perspective has served as the foundation for encryption schemes and many cryptographic protocols which are designed to protect against such external threats.

However, as the field of cryptography continues to evolve, it becomes increasingly important to explore alternative settings that challenge this traditional view. One notable example traces back to Yao's seminal work [Yao86] that introduced the *secure multi-party computation* (MPC),

a cryptographic framework that allows two or more parties to jointly compute a function over their respective inputs while maintaining the privacy of those inputs. In this context, the adversarial model shifts significantly; rather than merely considering external threats, we must account for the possibility that some participants may act maliciously or untrustworthily.

This requires the development of methods and techniques that can handle various types of adversarial behavior, whether it be from a single participant or a coalition of participants who may *collude* and may be *corrupted* before or during the protocol. This also requires some effort to (re)define the adversarial model and incorporating the dynamics of multiple parties, but helps to develop more robust systems that are better equipped to handle the complexities of real-world interactions.

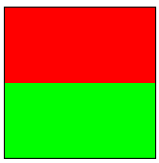
We refer to Section 2.2.3.5 for a formal overview of the *Universal Composability* (UC) model [Can01], a notable framework that gives strong security guarantees of protocols that may be used as a part of larger protocols, and hence can be safely *composed*.

1.1.3 Proof Systems and Zero-Knowledge

Traditional mathematical proofs are either self-evident or are based on established rules and axioms; moreover, it is something that can be written, and its correctness can be checked line-by-line.

As stated by Oded Goldreich [Gol95], the notion of proof in cryptography is arguably different: a cryptographic proof system is an *interactive protocol* where the prover seeks to convince the verifier of the truth of a claim. In the case of zero-knowledge proofs, an additional requirement is that the proof reveals nothing beyond the validity of the statement itself. This may seem paradoxical and counter-intuitive at first, thus we provide a simple example that may help to understand the concept.[1]

An example of a zero-knowledge proof. Imagine a scenario where a prover, who can differentiate colors, wants to convince a color-blind verifier that a particular page is not monochromatic, for example it has two colors like this:



The prover's goal is to convince the verifier that this page has two colors without transferring any additional information or "knowledge" (the colors themselves, in this case, or even the possibility of distinguishing them). Here's how the process unfolds.

1. The verifier begins by flipping a coin. The verifier performs this action on his own, not revealing the outcome of the coin flip to the prover. If the coin lands on heads, he will flip the page over; if it lands on tails, he will leave the page unchanged.

2. The prover examines the page. Since he knows what the page looked like before and can distinguish colors, he can determine whether the verifier flipped it or not. He checks if the page has been flipped and tells the verifier the outcome of the coin flip.

[1] This example is adapted from a talk of Shafi Goldwasser.

3. The verifier compares his own coin flip with the one guessed by the prover. If they match, he concludes that the prover could tell whether the page was flipped, which implies that there must be two colors on the page. Thus, he accepts the claim as true. Otherwise he rejects the claim.

If there are indeed two colors on the page, the prover can reliably follow this procedure, and the verifier will always accept the claim: this property is known as *completeness*.

However, if the page has a single color, the prover cannot determine if it was flipped or not. He might guess, resulting in a 50% chance of being correct. Thus, the probability that he convinces the verifier is at most 50%, which is due to luck, as the prover's claim of two colors is false: this property is known as *soundness*. The probability of this bad event can be reduced by repeating the process multiple times, thus reducing the soundness error up to a very small factor.

Since the prover does not reveal anything but the outcome of the coin flip, this proof is a *zero-knowledge* proof.

Proofs of knowledge. Imagine the scenario in which a user wishes to authenticate to a website. The user claims that the following statement is true: "there exists a password `pass` that is the correct password for the username `user`". In cryptographic terms, `pass` is referred to as the *witness* associated with the statement.

Since all registered users must have set up some password, and we want to enforce that only someone who really *knows* the password can successfully authenticate, proving the correctness of this claim is not enough. This is where a proof of knowledge (PoK) comes into play.

A PoK is a proof that allows the prover not only to convince the verifier that a statement is true, but also that he knows a valid witness associated with the statement, which in this case is the password that the user `user` used to register on the website.

If the PoK is also zero-knowledge, it means that while the verifier is convinced that the user does know the password, it gains no additional information about the witness itself. This solution is ideal in this setting because it additionally ensures that the user's password remains secret even if an eavesdropper observes the communication between the user and the website.

Non-interactive proofs. While in the above examples we have depicted the zero-knowledge cryptographic proofs as interactive protocols, where both parties can communicate in real-time, making them suitable for protocols like password authentication, proofs that do not involve an interaction between the prover and the verifier are also possible and known as *non-interactive zero-knowledge proofs* (NIZK). In non-interactive proofs, the prover generates a proof that can be sent in a single message to the verifier. This feature makes them attractive for a wide variety of applications, however a non-interactive proof is intrinsically transferable [GOS06b], which means that it should not be used in the setting of password authentication seen above: roughly speaking, the user `user`, knowing the correct password, could generate a *static*, non-interactive PoK and share it with other users, letting them authenticate to the website, without knowing the password themselves.

The Fiat-Shamir transform. In some interactive protocols, the messages of the verifier are truly random and independent of the messages of the prover: these protocols are said to be *public-coin* protocols. These protocols can be converted into non-interactive protocols by using the Fiat-Shamir transform [FS87], a heuristic method that replaces the random messages of the verifier with an invocation of a random oracle (RO) on the messages of the prover.

A primer on the security models and setups. We know (see [GO94]) that it is not possible to have NIZKs, for non-trivial "tasks", without making any computational assumptions (cf. Section 2.3), which is referred to as the Standard Model in cryptography (cf. Section 2.2.3.1). However, it is possible to construct NIZKs for a wide range of tasks by relaxing the security model or making additional assumptions.

A common setting in which NIZKs can be constructed for a wide class of tasks is when the prover and the verifier both have access to a common bitstring chosen by a trusted party: this model is referred to as the *CRS model*, introduced by Ivan Damgård [Dam00]. If the CRS needs to have some specific structure, we refer to it as a *structured CRS*, or simply SRS: this is usually to remark that the reference string has some *algebraic* structure. On the other hand, if the CRS is a random bitstring, we sometimes refer to it as the *uniform reference string*, or URS model. For some schemes, it is necessary to generate one CRS for each different task, while in other cases, a single CRS can be used for all tasks: this latter model is referred to as the *universal CRS*, in the sense that any task, up to a certain level of complexity, can be handled by a single CRS. Moreover, in some cases, it is possible to *update* the CRS without having to generate a new one, which is referred to as the *updatable CRS* model.

Some examples of NIZKs. The first NIZK was proposed by Blum, Feldman and Micali [BFM88], although in their construction the CRS cannot be reused for different statements (in this sense, their NIZK is called *bounded*) and additionally it limits the length of the statement, i.e., the difficulty of the task, that can be proved. This scheme was later improved by De Santis, Micali and Persiano [DMP90] to support proving many statements with the same CRS. Later on, Feige, Lapidot and Shamir [FLS90] showed how to construct NIZKs based on a more general class of assumptions, rather than just specific ones as in [BFM88, DMP90]. Efficient NIZKs were subsequently proposed [Dam93, KP98, BDP00] until the advent of *pairing-based* cryptography, which exploits special mathematical functions, called "pairings". A line of work, starting with the work of Groth, Ostrovsky, and Sahai [GOS06b, GOS06a], allowed for NIZKs for any NP language. This line of research culminated with the work of [GS08], which is simply referred to as the Groth-Sahai proofs, and which greatly improved the efficiency of NIZKs, although at the cost of restricting to a (powerful and meaningful) class of NP languages.

Proofs and arguments: a note on the terminology. Although in this informal introduction we use the terms rather interchangeably, proofs and arguments are different. An argument is a proof whose soundness is assumed only to hold against cheating provers that run in polynomial time, which means that a cheating prover cannot convince the verifier of a false statement *unless* he breaks some (presumably hard) cryptographic assumption. For this reason arguments are sometimes referred to as *computationally sound proofs*.

The toy example of the color-blind verifier above is in fact a proof. Later in this manuscript, we will give a formal definition of proof systems (cf. Definition 7.4.1). For the majority of the time, we will focus on non-interactive arguments though.

1.1.3.1 zkSNARKs

A particular case of interest is the class of zero-knowledge *succinct* and *non-interactive* arguments of knowledge, or simply zkSNARKs. The reason why zkSNARKs are so relevant from both practical and theoretical perspectives is that they combine the advantages of the (zero-knowledge) arguments of knowledge with their efficiency, i.e., the resources required to generate

and to verify the proof, but also their *size*. An active area of research on zkSNARKs has seen rapid progress in multiple aspects, such as efficiency [BCG+13, GGPR13, Gro10a, Gro16], security and versatility of their setups [BBHR19, GKM+18], and proof composition [BDFG21, BCMS20]. We defer a formal definition of zkSNARKs to Section 2.4.3.2 as it is beyond the scope of this introductive survey, but in what follows we give an overview of the major works in this field and the main ideas behind them.

SNARKs based on QSPs. Many works have tried to build SNARKs for *circuit satisfiability*: by this, we mean that a prover, given a circuit has to convince the verifier that it knows an assignment of its inputs that makes the output true. An interesting line of works has as a central starting point the framework based on quadratic span programs (QSP) introduced by Gennaro, Gentry, Parno and Raykova [GGPR13]: probably the two most known are Groth16 [Gro16], that is used in practice due to its very short proofs, and the scheme Pinocchio, proposed by Parno, Howell, Gentry and Raykova [PHGR13], based on the span programs for arithmetic circuits (QAP), a variant of QSP. The main idea behind QSP/QAP is to represent each gate of the circuit as a variable and to associate an equation with the logic of the circuit. Interestingly, a satisfying assignment for any circuit can be specified as a set of quadratic equations, hence the name *quadratic*. As a consequence, in these schemes the prover needs to convince the verifier that all the quadratic equations associated with the circuit are satisfiable.

These schemes are not universal as they require a specific CRS tailored to the circuit/task being proven.

SNARKs based on PCPs. Another class of SNARKs is the one based on the Probabilistically Checkable Proofs (PCP), which is a generalization of a proof system that allows the verifier to check the correctness of a proof by reading only some part of it; although it may seem somewhat counter-intuitive at first, the key idea behind PCPs is that the verifier can use the randomness and these proofs are *redundant*, and thus it is sufficient to check their well-formedness only in a few randomly chosen parts. Using this approach as starting point, several SNARKs have been proposed [Mic94, BCCT12, DFH12, BBHR18]. In particular, the work of Micali [Mic94] showed how to make non-interactive, by applying the Fiat-Shamir transform, the construction of Kilian [Kil92] in which the prover sends a commitment, i.e., a short and compressed digest (cf. Definition 2.4.3), of a PCP proof to the verifier and then the verifier checks the proof by asking for evaluations of the PCP proof in a few points.

It is worth noticing that the Micali construction is a special case of a process called *compilation*. A common approach to build zkSNARKs is to first construct a public-coin and interactive information-theoretic protocol that achieves the desired functionality in an idealized model and then remove the idealized component by *compiling* it into a zkSNARK via the use of a computationally-secure primitive, such as a commitment scheme and by applying the Fiat-Shamir transform to make the proof non-interactive. In this sense, the scheme of Micali can be seen as an example of *compilation* from PCPs to SNARKs.

SNARKs based on PIOPs. More recent works have focused on different and novel approaches to build the idealized component, in particular the Polynomial Interactive Oracle Proofs (PIOPs). In a PIOP, the prover sends polynomials that can be evaluated by the verifier. The commitment scheme used in the compilation is then a polynomial commitment scheme as it has to support the commitment of polynomials. We elaborate more on this specific compiler from PIOPs to zkSNARKs in Section 2.4.3.3 where we give a formal definition. We notice that many efficient constructions have emerged based on this paradigm, such as

Sonic [MBKM19], Marlin [CHM+20], Plonk [GWC19] and Basilisk [RZ21]. These are called pre-processing SNARKs since an CRS, although universal (and updatable) must be generated in advance. Other notable schemes that do not need a trusted setup, and are hence called *transparent*, are Aurora [BCR+19], STARK [BBHR18], and Spartan [Set20].

1.1.3.2 zkVMs

A popular approach to SNARKs is that of SNARKs for *Virtual Machines* (or SNARK VMs[2]), which at their heart consist of proving the execution of a computer program—expressed in a predetermined instruction set—over some CPU abstraction. This design has a number of attractive features: it makes available all the existing optimizing compilers for pre-existing instruction sets; it offers an excellent developer experience making SNARKs usable by anyone able to write a computer program [AST24, Tha24b]. Many SNARKs that are currently being deployed in practice follow this design pattern. Examples include the Cairo-VM [GPR21], the RISC Zero project [Zer], Scroll's Ceno [LZZ+24], Polygon Miden [Lab] and many others. Among these constructions, a notable example is Jolt [AST24], a SNARK for VMs that is based on the *lookup-singularity* approach [Whi22], which consists in reducing execution of opcodes in a VM to a series of table lookups. This approach has huge potential for adoption being simple, as well as easy to extend and to audit. It also leads to extremely fast provers (up to 2x faster than the current state of the art [Tha24a]).

1.2 Malleability

The Cambridge English Dictionary defines *malleability* as "the ability to be easily changed into a new shape". Aluminum, for example, can be effortlessly rolled into thin foils or shaped into various forms, making it highly useful for packaging and construction. Gold and silver exhibit similar malleable properties.

However, in certain structural applications, a highly malleable material may not withstand specific stresses or loads effectively, leading to deformation or failure. For this reason, reinforced concrete is often used as a substitute for more malleable materials in the construction of buildings and bridges. Similarly, while much of their original structure has been lost over the years, the stone foundations and columns of Greek temples that we see today have withstood the test of time.

Malleability in Cryptography. In cryptographic terms, malleability captures the possibility of efficiently modifying and transforming a cryptographic object, say an encrypted message or a proof, into a new valid one. A natural question may arise.

Is malleability a good or bad property in cryptography?

Keeping the analogy with the materials in the physical world, malleability is perceived as an undesirable property in a general-purpose cryptosystem from which we require some *solid foundation*, like in buildings, as it leads to issues and undesired attacks.

[2]Differently from how the expression zkVM is commonly used—in the sense of succinct and scalable arguments of knowledge for VMs which might or might not be zero-knowledge—in this by "zkVMs" we actually mean SNARKs for VMs *with* zero-knowledge. We will explicitly write "SNARKs for virtual machines" (or VMs) whenever we make no assumption on whether the underlying construction is zero-knowledge. We refer the reader to Section 5.8.1 for a formal definition of zkVMs.

However, there are also many applications where malleability is useful and desired; indeed, it is a crucial property in those settings where the possibility of being *shaped* into various forms solves specific problems. Thus, a more interesting question is:

What is the relationship between malleability and non-malleability?

In what follows, we explore some of the basic definitions and applications of malleable cryptographic objects such as public key encryption schemes and proof systems.

1.2.1 Malleability and non-malleability in PKE schemes

Since a formal definition is quite involved, we prefer giving a rough intuition of what malleability means for a PKE. A malleable PKE allows for modifications to ciphertexts, in the sense that one can modify an encryption c of some unknown message $\texttt{msg}$, and come up with a ciphertext c' that is an encryption of a message $\texttt{msg}'$ that is related in some known way to $\texttt{msg}$: essentially, the change on the ciphertext *predictably* applies to the corresponding plaintext message too.

Homomorphic Encryption. Let say that anyone can transform an encryption of a message $\texttt{msg}$ into a *valid* encryption of $f(\texttt{msg})$, where f is some function. Such schemes are known as *homomorphic* encryption schemes and are valuable because they allow for secure processing of sensitive data, as we show with the following example.

Imagine a hospital that collects sensitive patient data, such as test results and personal health information. The hospital wants to analyze this data for a research study, but it has to ensure that patient privacy is maintained.

Even if the hospital encrypts the patient data, the analysis must still be performed on the "plaintext" data, which potentially exposes to the risk of unauthorized access to it.

Using homomorphic encryption, researchers can perform their calculations directly on the encrypted data collected by the hospital, without ever the need of decrypting it. In this case, the function f could be as simple as some statistics (sum, mean, median) or a complex machine learning model. Then, patient data remains encrypted throughout the entire process.

CCA security. Depending on the context, however, the possibility of *altering* (or "tampering") messages may lead to miscommunication or even cause serious consequences, e.g., in a financial transaction or a legal agreement. To mitigate this serious vulnerability, Rackoff and Simon [RS92] formalized what is still now considered the *standard* notion of security for PKE schemes, called Security against chosen-ciphertext attacks (CCA). This security definition essentially prevents all the malleability attacks and is achieved by several schemes.

1.2.2 Malleability and non-malleability in proof systems

Chase *et al.* [CKLM12] noticed that malleable proofs can boost the efficiency of some protocols, such as secure electronic voting systems. Despite useful, malleability for cryptographic proofs may be an opportunity for attacks, as we elaborate hereafter.

Knowledge-soundness. The basic security notion of a PoK is *knowledge-soundness* (cf. Section 2.4.3.1): informally speaking, it guarantees that, in isolation, a prover producing a valid proof must know the corresponding witness. Looking back at the example in Section 1.1.3, a proof system is *knowledge-sound* if it is very unlikely that a user who successfully authenticates to the website does not know the secret password.

Simulation extractability. There exist real-world deployments and cryptographic applications of NIZKs and zkSNARKs that require a stronger property called *simulation extractability* [Sah99, DDO+01, GM17], that can be expressed in the following informal way, quoting [DDO+01]:

> "*Whatever an adversary can prove after seeing polynomially many NIZK proofs for statements of its choosing, it could have proven without seeing them, except for the ability to duplicate proofs*"

Intuitively, this notion considers attackers that can see proofs for some statements and may use this information in order to produce a proof for some other statement without knowing the witness. Interestingly, simulation extractability implies that proofs are *non-malleable* [DDN91]. For example, this prevents that an eavesdropper that can see many users successfully authenticating to a website can use this information to authenticate himself to the website, without knowing the password.

1.2.3 Controlled malleability

While malleability seems an attractive feature for a cryptosystem, it seems to be in conflict with strong security guarantees.

However, under a relaxed but still meaningful security definition, we can have a proof system or an encryption scheme that is both malleable, for a class of operation, and non-malleable to a satisfactory extent. We refer to this notion as *controlled malleability*. This concept was formalized by Chase *et al.* [CKLM12] in the context of NIZKs, but it can be extended to other cryptographic primitives and schemes.

1.3 Research Questions

In this manuscript, we explore a number of research questions that are relevant to the malleability and the non-malleability of important cryptographic primitives and schemes, and we aim to provide a cohesive narrative that incorporates some recent developments that have been published concurrently or after some of the work in this manuscript.

We analyze some of the theoretical foundations that underpin malleability and its implications for cryptographic security: along the way, we either propose new schemes or we revisit the security analysis of existing constructions, and we investigate the mechanisms that can transform malleable schemes into non-malleable counterparts. By examining the boundaries and connections between malleability and non-malleability, we aim to provide a deeper understanding of how they can actually coexist in robust and efficient cryptographic protocols.

List of publications. The results presented throughout this manuscript have been published in [FR22] (co-authored with Antonio Faonio), [FHR23] (co-authored with Antonio Faonio and Dennis Hofheinz), [FFK+23] (co-authored with Antonio Faonio, Dario Fiore, Markus Kohlweiss and Michal Zajac) [FFR24] (co-authored with Antonio Faonio and Dario Fiore), [CFR25] (co-authored with Matteo Campanelli and Antonio Faonio).

A recent result [BCC+24] (co-authored with Christian Badertscher, Matteo Campanelli, Michele Ciampi and Luisa Siniscalchi) on the non-malleability of SNARKs has not been included in this thesis as it is under review.

Note to the reader. In the next pages, we elaborate on the main research questions and the contributions that are developed in this thesis. Especially when we will delve into the state of the art, the discussion becomes more technical in nature. This shift is intended for readers who are already familiar with the field, as it builds upon established concepts and methodologies. For those who may need to familiarize themselves with certain technical aspects, we recommend consulting Chapter 2, which provides a foundational overview of the relevant theories and terminologies.

1.3.1 Simulation-extractable zkSNARKs

Most zkSNARKs in the literature are only proven to be knowledge-sound. In some cases, this is due to the fact that their proofs may indeed be malleable, e.g., as in [Gro16]. In other cases, the lack of a proof is because it is a challenging task that does not follow via a straightforward extension of the knowledge-soundness proof, and hence may require a different approach.

The state of simulation-extractable zkSNARKs. Jens Groth and Mary Maller give a simulation-extractable zkSNARK that consists of only 3 group elements [GM17], but their construction is neither universal nor updatable.

The work of Ganesh, Orlandi, Pancholi, Takahashi and Tschudi [GOP+22] shows that Bulletproofs [BBB+18] are non-malleable in the Algebraic Group Model (see Section 2.2.3.4). Quang Dao and Paul Grubbs show that Spartan [Set20] and Bulletproofs are non-malleable even without the AGM [DG23]. Both these works extend the framework introduced by Faust, Kohlweiss, Marson and Venturi in [FKMV12] to the Fiat-Shamir transform applied to multi-round interactive arguments. On a similar path, the work of Ganesh, Khoshakhlagh, Kohlweiss, Nitulescu and Zajac [GKK+22] shows non-malleability for Plonk [GWC19], Sonic [MBKM19] and Marlin [CHM+20]. Both [GKK+22, GOP+22] show that interactive arguments can be simulation-extractable after applying the Fiat-Shamir transform. In particular, their approach consists into defining new properties: *trapdoor-less zero-knowledge*, i.e., zero-knowledge where the simulator does not rely on the SRS's trapdoor but on the programmability of the random oracle, and *unique response*, i.e., at some point of the protocol, the prover becomes a deterministic algorithm. Crucially, these properties need to be proven *on a case-by-case basis*; namely, for each candidate SNARK (even if resulting from a generic compiler) one needs additional effort to show that it is simulation extractable. Furthermore, [GKK+22] relies on rewinding-based soundness, which is a non-standard notion of soundness, and rewinding which make the extractor success dependent on the probability of the adversary returning an acceptable proof; and [GOP+22] relies on the AGM.

1.3.1.1 A general framework for simulation-extractable zkSNARKs

Can we find some simple conditions that guarantee non-malleability of zkSNARKs?

At first, one may wonder whether the natural compilation from PIOPs to zkSNARKs leads to simulation-extractable zkSNARKs if the polynomial commitment is simulation-extractable. Despite possible, this result would not capture existing (instantiations of) zkSNARKs, because the popular KZG polynomial commitment [KZG10], that leads to some of the most efficient instantiations, is homomorphic.

To address this, we investigate more deeply the malleability of the KZG polynomial commitment in Chapter 3. We then show that the natural compilation from PIOPs to zkSNARKs leads to simulation-extractable zkSNARKs as long as the polynomial commitment possesses some special form of simulation extractability (or controlled-malleability), which we will show the KZG polynomial commitment has, and the PIOP satisfies some simple conditions, met by popular schemes such as Plonk [GWC19] and Marlin [CHM+20].

Our results on the compiler do not directly rely on the AGM, do not require the protocol to be trapdoor-less zero knowledge or have the unique-response property and use more standard notion of soundness, specifically the *state restoration soundness* of the PIOPs [BCS16], which guarantees that the argument system is sound even if the adversary can *rewind* to some previous state the verifier.

1.3.1.2 Real-World zkSNARKs

Our result in Chapter 3 provides a framework for the study of the PIOP-to-zkSNARK compilation. However, the schemes that offer the best performance and are eventually implemented in software libraries depart from the ones obtained through that compilation process. In particular, *real-world* versions of schemes such as Marlin or Plonk make use of an optimization, the *linearization trick* (also known as Maller's optimization) [GWC19, OL], that leverages the homomorphic properties of the KZG polynomial commitment to reduce the number of field elements in the proof. This optimization though changes the zkSNARK verification algorithm in a way that escapes the security analysis of Chapter 3; a similar limitation holds for the work of Kohlweiss, Pancholi and Takahashi [KPT23] that, by extending the techniques of [GKO+23], shows how to compile Algebraic Holographic Proofs (AHPs) to non-malleable zkSNARKs.

Can we prove non-malleability for optimized real-world zkSNARKs?

In Chapter 4 we show how to resolve the limitations above, and we give the first proof of simulation extractability of the "real world" optimized versions of zkSNARKs which include Plonk [GWC19], Marlin [CHM+20], Lunar [CFF+21] and Basilisk [RZ21]. Along the way, we give the first formal analysis of the linearization trick, in particular its knowledge-soundness and simulation extractability in the AGM with oblivious sampling (cf. Section 2.2.3.4).

1.3.1.3 SNARKs for Virtual Machines

None of the previous results on the simulation extractability of zkSNARKs cover the case of zkVMs. One way to achieve simulation extractability for a zkVM is to compose it with another zkSNARK, i.e., we could use a zkSNARK to prove the knowledge of a valid zkVM proof (e.g., the recent work Testudo [CGG+23] composes Spartan with Groth16 [Gro16], and some folding-based schemes such as Nova [KST22] follow this approach). Of course, if this zkSNARK is also simulation-extractable, then it seems we get the maximum result with the minimum effort.

Despite viable, this approach of "adding" zero-knowledge by composition has some theoretical and practical drawbacks. In particular, it would require representing the verifier of the zkVM in a format which may be cumbersome and partially limit the benefits of the improved auditability of zkVMs like Jolt. Furthermore, this *arithmetization* procedure incurs in a direct random oracle instantiation that hence becomes public to the adversary, which may lead to insecure schemes [CGH98].

Since zkVMs are behind the design of deployed systems with non-malleability requirements, this remains an urgent open problem.

What can we say about the non-malleability of zkVMs?

In Chapter 5 we address this problem by showing that a modular zkVM inspired by Jolt achieves simulation extractability under minimal assumptions. We make a step further, and we explore the more general setting of the composition of zkSNARKs.

1.3.2 Re-randomizable PKE

An interesting class of malleable public key encryption schemes are those that are *re-randomizable*. By re-randomizable we mean that the ciphertext c of a message `msg` can be transformed into a new ciphertext c', that looks like a fresh ciphertext, but decrypts to the *same* message. This feature is useful to implement *proxy re-encryption* schemes [BBS98], which allow a proxy server to transform a ciphertext computed under one's public key into one that can be decrypted by another secret key, and is met by popular PKE schemes, such as the ElGamal PKE [ElG84].

Re-randomization is in fact a simple form of malleability. Typically, we would like to offer security against malleability attacks on the encrypted message: an eavesdropper should not be able to *transform* a ciphertext of a message `msg` into a ciphertext of a different message $\texttt{msg}'$ (cf. Section 1.2.1). Ideally, we would like to have a scheme that can be re-randomized, so that we can use its re-randomization to implement protocols like *proxy re-encryption*, but at the same time we would like to prevent all the other malleability attacks.

Rand-RCCA PKE. To this end, a relaxed version of CCA security for PKEs, dubbed Replayable CCA security (RCCA), was introduced in [CKN03] and later extended by Jens Groth [Gro04] to the Re-randomizable Replayable CCA (Rand-RCCA) setting. Constructing Rand-RCCA-secure PKE has been generally considered a difficult problem and was posed as an open problem in [CKN03]. The scheme proposed by Groth [Gro04] is secure in the Generic Group Model (cf. Section 2.2.3.3) and the ciphertext size expansion is as large as the bit-length of the plaintext. Prabhakaran and Rosulek showed how to obtain Rand-RCCA schemes under minimal assumptions [PR07]. Chase *et al.* [CKLM12] proposed a new Rand-RCCA-secure PKE scheme by using malleable NIZKs, and their work has been later improved by Libert, Peters and Qian [LPQ17], although it still incurred high computational costs and a large ciphertext size. More efficient schemes, with extremely short ciphertexts, have been proposed by Faonio *et al.* [FFHR19, FF20].

1.3.2.1 Applications

We investigated the potential applications of Rand-RCCA PKE schemes, in particular in the context of verifiable mix-nets.

A Mix-net (or mixing network) is a protocol introduced by David Chaum [Cha81] that allows a set of senders to send messages anonymously, and finds applications in different domains, including anonymous e-mail, anonymous payments and electronic voting, to name a few. A natural and simple design of mixing networks are re-encryption mix-nets [PIK94] in which each mixer first decrypts and then *freshly* encrypts every ciphertext: this operation can be performed even publicly using re-randomizable encryption schemes such as ElGamal. The process of re-randomizing and randomly permuting ciphertexts is also called *shuffle*.

The use of zero-knowledge arguments to prove the correctness of a shuffle was first suggested by Sako and Kilian [SK95]. The first proposals used expensive *cut-and-choose*-based zero-knowledge techniques [Abe98, SK95]. Abe *et al.* removed the need for cut-and-choose by proposing a shuffle based on permutation networks [Abe99, AH01]. Furukawa and Sako [FS01] and independently Neff [Nef01] proposed the first zero-knowledge shuffle arguments for ElGamal ciphertexts that achieve a complexity linear in the number of ciphertexts. These results have been improved by Wikström [Wik09], and later Terelius and Wikström [TW10], who proposed arguments where the proof generation can be split into an offline and online phase. These protocols have been implemented in the Verificatum library [Wik10]. Groth and Ishai [GI08] proposed the first zero-knowledge shuffle argument with sublinear communication. Bayer and Groth gave a faster argument with sublinear communication in [BG12].

Most of the research effort for improving the efficiency of mix-nets has been so devoted to improving the efficiency of shuffle arguments, and in this sense the work of Faonio *et al.* [FFHR19] is a notable exception. Roughly speaking, their work proposes as main building block a Rand-RCCA PKE scheme rather than a CPA-secure PKE scheme such as ElGamal to achieve faster and simpler instantiations of the proof systems required for the shuffle. To make the protocol verifiable, however, they require a Rand-RCCA PKE scheme that is publicly verifiable and this makes their construction less efficient.

Can we improve the efficiency of the protocols that rely on Rand-RCCA PKE schemes?

In Chapter 6 we answer affirmatively. We revisit the mix-net design of [FFHR19], and we give a more efficient instantiation for the mix-net protocol: we show a protocol that is in fact *verifiable* even if it is based on a non publicly-verifiable, and more efficient, Rand-RCCA scheme.

1.3.2.2 Tight Security

George Orwell, in his book Animal Farm, wrote that: "All animals are equal, but some are more equal than others".

In a way, we can also say that all PKE schemes are secure, but some are more secure than others. Indeed, while it is often clear that a PKE scheme is secure, it is often unclear how much *concrete* security it really offers when it is used in the wild. Bellare, Boldyreva and Micali, in their seminal work [BBM00], investigated how tight the security of an encryption scheme translates to the trust that we have with respect to the cryptographic assumption that it relies on.

In more detail, a tight security reduction ensures that for any attack on the PKE scheme, there exists an attack on the assumption that is similar both in terms of complexity (i.e. the running time, the space required, etc.) and success probability.

Thus, in the setting of tight security reductions, the number of ciphertexts considered by the security definition matters. By now, many CCA-secure PKE schemes (cf. Section 1.2.1) have been proved to have tight security in the multi-ciphertext and multi-user setting: some notable examples are the works of [GHKW16, GHK17, HLLG19, Hof17, LJYP14, LPJY15]. However, tight security in the context of Rand-RCCA security has not been studied, although in particular the above Rand-RCCA use cases feature numerous ciphertexts or users.

Is there any Rand-RCCA PKE scheme with tight security?

In Chapter 7 we introduce the notion of multi-user and multi-ciphertext Rand-RCCA PKE, and we give the first construction of such a PKE scheme with a tight security reduction to a standard computational assumption.

1.4 Organization of the manuscript

This thesis is organized into eight chapters that, besides the introduction, are structured as follows.

- Chapter 2 introduces the notation used throughout the manuscript and formally introduces some mathematical and computational concepts, as well as some cryptographic assumptions and schemes.
- Chapter 3 introduces the framework of policy-based simulation extractability, and shows how to obtain simulation-extractable zkSNARKs from suitable polynomial commitments.
- Chapter 4 extends this framework, analyzes the security of the linearization trick, and shows how to obtain optimized simulation-extractable zkSNARKs.
- Chapter 5 focuses on the simulation extractability of zkVMs and, more in general, studies the setting of modular commit-and-prove zkSNARKs.
- Chapter 6 shows how to improve the efficiency of mix-nets based on Rand-RCCA PKE schemes.
- Chapter 7 initiates the study of tight security in the Rand-RCCA setting.
- Chapter 8 draws final thoughts and outlines some of the questions that remain open.

Chapter 2

Background

In this chapter, we introduce the notation used throughout the manuscript. We recall some standard mathematical and computational concepts, as well as some number-theoretic and cryptographic assumptions. Most of the information is rather usual and would look familiar to a reader with a background in cryptography, thus it can be easily skimmed through.

2.1 Notation and Technical Preliminaries

For an integer $n \geq 1$, we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. Calligraphic letters denote sets, while set sizes are written as $|\mathcal{X}|$. Lists are represented as ordered tuples, e.g. $L := (L_i)_{i \in [n]}$ is a shortcut for the list of n elements $(L_1, \ldots, L_n)$. To get a specific value from a list, we also use the "dot" notation; e.g., we use $L.b$ to access the second element of the list $L = (a, b, c)$. The difference between lists and vectors is that elements of vectors are of the same type.

For any bit string $\tau \in \{0,1\}^*$, we denote by $\tau[i]$ the i-th bit of τ and by $\tau_{|i}$ the bit string comprising the first i bits of τ.

2.1.1 Mathematical notation and basic definitions

Cyclic Groups. A cyclic group $\mathbb{G}$ of order q is a (commutative) finite group that can be generated by a single element g, that we call the *generator*, namely $\mathbb{G} = \{1, g, g^2, \ldots, g^{q-1}\}$, where 1 denotes the identity element. When the order and the generator are clear from the context, we may omit them. Also, we assume that the multiplication over $\mathbb{G}$ is an efficient operation, so given g and $x \in \mathbb{Z}_q$, we can efficiently compute g^x.

Bilinear Groups. A bilinear group $\mathbb{G}$ is a tuple $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P_1, P_2)$, where $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are groups of prime order q, the elements P_1, P_2 are generators of $\mathbb{G}_1, \mathbb{G}_2$ respectively, and $e \colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficiently-computable non-degenerate bilinear map.

We can categorize a scheme's usage of the bilinear map into 3 standard categories:

Type-1: if $\mathbb{G}_1 = \mathbb{G}_2$. These groups are also called *symmetric*

Type-2: if there is an efficiently computable homomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$

Type-3: f there is no efficiently computable homomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$. These groups are also called *asymmetric*

Whenever possible, elements in $\mathbb{G}_i, i \in \{1, 2, T\}$ are denoted in implicit (additive) notation as $[a]_i := aP_i$, where $P_T := e(P_1, P_2)$. Every element in $\mathbb{G}_i$ can be written as $[a]_i$ for some $a \in \mathbb{Z}_q$.[1] Given $a, b \in \mathbb{Z}_q$ we distinguish between $[ab]_i$, namely the group element whose discrete logarithm base $\mathcal{P}_i$ is ab, and $[a]_i \cdot b$, namely the execution of the multiplication of $[a]_i$ and b, and $[a]_1 \cdot [b]_2 = [a \cdot b]_T$, namely the execution of a pairing between $[a]_1$ and $[b]_2$. We do not use the implicit notation for variables, e.g. $\mathsf{c} = [a]_1$ indicates that c is a variable name for the group element whose logarithm is a.

Vectors and Matrices. We extend the pairing operation to vectors and matrices (of elements in bilinear groups) as $e([\vec{A}]_1, [\vec{B}]_1) = [\vec{A}^\top \cdot \vec{B}]_T$ and $e([y]_1, [\vec{A}]_1) = [y \cdot \vec{A}]_T$.

Let $span(\vec{A})$ denote the linear span of the columns of $\vec{A}$. Given a set of vectors $\mathcal{V}$ in some vector space over $\mathbb{Z}_q$, $span(\mathcal{V})$ denotes its linear span.

Tensor Product. We define the transformation T that maps a matrix $\vec{A} = (\vec{a}_1, \ldots, \vec{a}_n) \in \mathbb{Z}_q^{n \times m}$ to the vector $T(\vec{A}) = (\vec{a}_1^\top, \ldots, \vec{a}_n^\top)^\top \in \mathbb{Z}_q^{n \cdot m}$. Namely, the transformation concatenates the columns of $\vec{A}$ to form a vector of length $n \cdot m$. We define the tensor product between two vectors $\vec{a}, \vec{b}$ to be $\vec{a} \otimes \vec{b} := T(\vec{a} \cdot \vec{b}^\top)$. Similarly, given $\vec{A} \in \mathbb{Z}_q^{n \times m}$ and $\vec{B} \in \mathbb{Z}_q^{n' \times m'}$, we define the vector product between two matrices $\vec{A} \otimes \vec{B}$ to be the matrices:

$$\left(T(\vec{a}_1\vec{b}_1^\top), \ldots, T(\vec{a}_1\vec{b}_{n'}^\top), \ldots, T(\vec{a}_n\vec{b}_1^\top), \ldots, T(\vec{a}_n\vec{b}_{n'}^\top)\right) \in \mathbb{Z}_q^{n \cdot n' \times m \cdot m'}.$$

We can show the following property:

$$(\vec{A} \cdot \vec{R}) \otimes (\vec{B} \cdot \vec{S}) = (\vec{A} \otimes \vec{B}) \cdot (\vec{R} \otimes \vec{S}) \tag{2.1}$$

Multilinear extensions. For any function $f\colon \{0,1\}^\ell \to \mathbb{F}$, there exists a unique ℓ-variate multilinear polynomial $\widetilde{f}$ such that $\widetilde{f}(x) = f(x)$ for all $x \in \{0,1\}^\ell$. We refer to $\widetilde{f}$ as the multilinear extension of f. For a vector $\vec{a} \in \mathbb{F}^n$, where n is a power of 2, we similarly define the multilinear extension $\widetilde{a}\colon \mathbb{F}^{\log n} \to \mathbb{F}$ as follows: we interpret a in the natural way as listing all n evaluations of a function with domain $\{0,1\}^{\log n}$, and define $\widetilde{a}$ to be the multilinear extension of this function.

2.1.2 The Schwartz-Zippel Lemma

Often the proofs in this document exploit the following basic property of polynomials, discovered independently by Richard DeMillo and Richard J. Lipton in 1978, and later by Jack Schwartz and Richard Zippel. Although DeMillo and Lipton's version was shown a year prior to the other versions, the lemma is commonly known as the Schwartz-Zippel lemma.

Lemma 2.1.1 (Schwartz-Zippel Lemma)**.** *Let $\mathbb{F}$ be any field, and let $f\colon \mathbb{F} \to \mathbb{F}$ be a nonzero polynomial of degree d. Then on any finite set $\mathcal{S} \subseteq \mathbb{F}$:*

$$\Pr[f(x) = 0 \mid x \leftarrow\!\$\, \mathcal{S}] \leq \tfrac{d}{|\mathcal{S}|}$$

The lemma can be generalized to multivariate polynomials whose total degree is d.

[1] However, it is in general hard to compute a from $[a]_i$, cf. Section 2.3.1

2.1.3 Algorithms

PPT and EPT algorithms. A PPT algorithm is a probabilistic algorithm whose running time is bounded by a polynomial in his input size. Sometimes we simply refer to PPT algorithms as *efficient* algorithms.

Conversely, an EPT algorithm is a probabilistic algorithm whose *expected* running time (over the choice of the random coins) is bounded by a polynomial in his input size. An EPT algorithm may have a running time that varies significantly based on the random choices it makes, but on average, the running time across all possible random executions is polynomial.

We write $y \leftarrow \mathrm{A}(x; r)$ to denote the fact that the algorithm A on input x and random coins r outputs a value y. Sometimes, when we do not need to explicitly specify the random coins r, we simply write $y \leftarrow\!\!\$\ \mathrm{A}(x)$.

2.2 Provable security

2.2.1 Unbounded security

The notion of *unbounded security*, originated with the work of Claude Shannon [Sha49], aims to designing primitives with provable security against adversaries whose computational resources are infinite. Intuitively, nobody is able to breach the security of the system because there is never enough *information* to succeed.

A classical example of this is the One-time Pad encryption scheme that dates back at least to Frank Miller in 1882 and patented by Gilbert Vernam in 1919, that is provably "unbreakable" because the ciphertext provides no information about the message, regardless of the computational power of the attacker.

2.2.2 Computational Security

Unbounded security provides strong theoretical guarantees, but it comes at the cost of practicality, efficiency, and usability in real-world applications.

Most cryptographic schemes employed today are designed with bounded security assumptions, where security is based on computational hardness assumptions (see Section 2.3), such as the difficulty of factoring large integers or solving discrete logarithms over some group, rather than absolute guarantees. We notice that, given enough computational power or enough time, these *hard* problems can be solved: in particular, one may employ a *brute-force attack* trying to solve a problem by enumerating every possible solution until finding a good one. This motivates a careful and rigorous approach to quantify the *finite* amount of computational power (e.g., time or space) and the type of the adversary.

Security Parameter. First, we introduce the notion of security parameter, that we denote by λ throughout the manuscript. Roughly, a cryptosystem provides λ bits of security if it requires 2^λ operations to be broken.

When analyzing the security of a scheme, we consider adversaries efficient w.r.t. the security parameter, in the sense that their running time (i.e., their number of simple operations) is bounded by some polynomial in λ. The algorithms of the schemes we consider are parameterized by the security parameter too, e.g., the key generation algorithm of a scheme takes as additional

input 1^λ to specify this dependency. It is worth noting that we sometimes omit the security parameter to ease the notation when it is clear from the context.

Negligible Functions. In the setting of computational security, a cryptosystem is considered secure if an efficient attacker can break it at most with some "small" probability, namely something that approaches zero faster than the reciprocal of any polynomial in λ, as we formalize hereafter.

Definition 2.2.1 (Negligible Function)**.** *We say that a function f is* negligible *in λ, and we write $f \in \mathsf{negl}(\lambda)$, if for every $c \in \mathbb{N}$ there is an integer λ_c such that $f(\lambda) \leq \lambda^{-c}$ for all $\lambda \geq \lambda_c$.*

2.2.3 Security Models

2.2.3.1 The Standard Model

The bare minimum computational (security) model that we can consider is one in which schemes can be proven secure using only complexity assumptions. This fundamental model is referred to as the *Standard Model* since the only limitation we put to the attackers is that they have some finite time and computational power available.

It is worth noting that, despite being the *weakest* computational security model available, in the sense that the adversaries are somehow the strongest that we can consider among the computationally bounded ones, security proofs are difficult to achieve in this setting. This justifies the need for introducing some idealized models, in which cryptographic primitives are replaced by idealized versions, or we put some additional and meaningful restriction to the class of adversaries.

2.2.3.2 The Random Oracle Model

One of the most well-known models is the *Random Oracle Model* (ROM) [BR93]. Roughly speaking, this model assumes the existence of a *truly random* function that all the parties involved in a protocol have oracle access to.

We notice that if we need to employ a scheme that is secure in the ROM, we cannot store the gigantic truth table of a truly random function, thus we need to replace every occurrence of the RO with a function whose output "looks like random", but that has an efficient and compact representation: an ideal candidate for this is a cryptographically-secure hash function, such as the SHA family of functions. The schemes proven secure in the ROM are widely used in practice because, very often, they are the most (or even the only) efficient ones we are aware of.

However, the instantiation of the RO leads to schemes that are only heuristically-secure. It is known that a direct instantiation of the RO results in insecure schemes [CGH98], even though these counterexamples seem somehow artificial and contrived [KM15].

2.2.3.3 The Generic Group Model

The *Generic Group Model* (GGM) is an idealized cryptographic model in which the adversary is given access to a randomly chosen encoding of a group: in particular, the adversary can

execute group operations (and pairings, if needed) by querying a suitable oracle that takes as input two encodings of group elements and outputs an encoding of a third element.[2]

This model is named generic after the fact that the algorithms do not exploit any special structure of the representation of the group elements and can thus be applied in any cyclic group.

Some important algorithms such as the Pohlig-Hellman algorithm [PH78] or the Pollard's rho algorithm [Pol78] are generic. However, many algorithms of practical interest are in fact not generic, e.g., the index-calculus algorithm is applicable only over groups in which the elements are represented as integers. This justifies the need for introducing a more realistic model, in which the adversary can exploit the structure of the group.

Moreover, the GGM suffers from some of the same problems as the ROM: it has been shown that there exist cryptographic schemes which are provably secure in the generic group model but which are trivially insecure once the random group encoding is replaced with an efficiently computable instantiation of the encoding function [Den02].

2.2.3.4 The Algebraic Group Model

The *Algebraic Group Model* (AGM) is a computational model introduced by Fuchsbauer, Kiltz and Loss [FKL18] in which all adversaries are modeled as algebraic, as we define hereafter.

Definition 2.2.2 (Algebraic algorithm, [FKL18])**.** *An algorithm $\mathcal{A}$ is* algebraic *if for all group elements* z *that $\mathcal{A}$ outputs (either as returned by $\mathcal{A}$ or by invoking an oracle), it additionally provides the representation of* z *relative to all previously received group elements. That is, if* elems *is the list of group elements that $\mathcal{A}$ has received so far, then $\mathcal{A}$ must also provide a vector $\vec{r}$ such that* $\mathsf{z} = \langle \vec{r}, \mathsf{elems} \rangle$.

This simple, yet natural, restriction makes the AGM stronger than the Standard Model, but weaker than the Maurer's GGM. One of the main benefits of this model over the GGM is the ability to capture adversaries that can exploit the (algebraic) structure of the group.

The AGM with Oblivious Sampling. In the AGM it is forbidden outputting a group element sampled obliviously, i.e., without knowing a valid representation. However, in many groups it is easy to sample group elements obliviously. In the case of elliptic curve groups, this is a concrete possibility due to admissible encodings, namely efficiently-computable functions that map elements of $\mathbb{Z}_q$ to an elliptic curve point whose discrete logarithm is unknown. Lipmaa, Parisella and Siim [LPS23] introduced a more realistic variant of the AGM, denoted as the AGM with Oblivious Sampling (AGMOS), that gives the algebraic adversary oracle access to an oblivious sampling procedure that returns group elements without revealing their discrete logarithm.

2.2.3.5 The Universal Composability model

The last model that we analyze is the *Universal Composability* (UC) model, introduced by Ran Canetti [Can01]. In this section, we briefly review some basic notions of this framework. One of the fundamental notions of the UC framework is the so-called adversarial environment, which,

[2] There are actually two generic group models in the literature, Shoup's [Sho97b] and Maurer's [Mau05], and we will mostly refer to the latter one.

roughly speaking, could be seen as an *interactive* distinguisher. The environment Z provides the inputs to all the parties of the protocols, and schedules both the order of the messages in the networks and the order in which the parties are activated. Also, Z decides which party to corrupt. We consider static corruption model where the set of corrupted parties is decided by Z before the protocol starts. Let consider the following executions:

$\mathsf{Real}_{\Pi,\mathcal{A},\mathsf{Z}}(\lambda)$**:** run an interaction involving the adversary $\mathcal{A}$, the environment Z and the honest parties. Z generates inputs for honest parties and activates them, the honest parties run the protocol Π, and give their outputs to Z. Finally, Z outputs a value that is taken as the output of $\mathsf{Real}_{\Pi,\mathcal{A},\mathsf{Z}}(\lambda)$.

$\mathsf{Ideal}_{\mathcal{F},\mathcal{S},\mathsf{Z}}(\lambda)$**:** run an interaction involving the simulator $\mathcal{S}$ and the environment Z. When Z generates the input for an honest party, the input is passed directly to the functionality $\mathcal{F}$, and the corresponding output is given to Z on behalf of that honest party. The output value of Z is taken as the output of $\mathsf{Ideal}_{\mathcal{F},\mathcal{S},\mathsf{Z}}(\lambda)$.

Without loss of generality, the environment's final output can be just a single bit. The bit represents the environment's "guess" of whether it run in the real or ideal world.

For the sake of modularity, it is useful to design a protocol Π that realizes a functionality $\mathcal{F}$ that makes use of other ideal functionalities, e.g., the functionality $\mathcal{G}$. We can define the $\mathcal{G}$-hybrid world, where the parties of Π also interact with the additional functionality $\mathcal{G}$.

$\mathcal{G}\text{-}\mathsf{Hybrid}_{\Pi,\mathcal{A},\mathsf{Z}}(\lambda)$**:** run an interaction involving adversary $\mathcal{A}$, environment Z and the ideal functionality $\mathcal{G}$. When Z generates an input for an honest party, the honest party runs the protocol Π, and gives its output to Z. The parties and the adversary can interact, by sending and receiving messages, with the ideal functionality $\mathcal{G}$. Finally, Z outputs a value.

We are now ready to give the main security definition for the universal composability model.

Definition 2.2.3. *A protocol Π UC-realizes an ideal functionality $\mathcal{F}$ with setup assumption $\mathcal{G}$ if for all PT real-world adversaries $\mathcal{A}$ there exists a PPT simulator $\mathcal{S}$ such that, for all environments PT Z:*

$$|\Pr\left[\mathcal{G}\text{-}\mathsf{Hybrid}_{\Pi,\mathcal{A},\mathsf{Z}}(\lambda)=1\right]-\Pr[\mathsf{Ideal}_{\mathcal{F},\mathcal{S},\mathsf{Z}}(\lambda)=1]|\in\mathsf{negl}(\lambda)$$

Let Π be a protocol that securely realizes an ideal functionality $\mathcal{F}$ in the $\mathcal{G}$-hybrid world, and let Σ be a protocol that securely realizes the ideal functionality $\mathcal{G}$. Then, *composing* Π and Σ, i.e., replacing every invocation of $\mathcal{G}$ with a suitable invocation of Σ, results in a secure protocol for $\mathcal{F}$.

When specifying an ideal functionality, we use the "delayed outputs" terminology adopted in [Can01]: when a functionality $\mathcal{F}$ sends a public delayed output y to a party $\mathcal{P}$, we mean that y is first sent to the simulator $\mathcal{S}$ and then forwarded to $\mathcal{P}$ only after an acknowledgment by $\mathcal{S}$.

For the sake of simplicity, we assume that all the parties and all the ideal functionalities have an implicit public parameter pp hardcoded. We can think of pp as being the description of a cryptographic group or some other publicly-available information such as the description of a cryptographic hash function. We slightly tweak the definitions of both the hybrid and the ideal world to include such public parameters; specifically, we consider that the ideal world (resp. hybrid world) samples $\mathsf{pp}\leftarrow\$\,\mathsf{Setup}(1^\lambda)$ and passes along this information, together with

the security parameters, to all the ITMs involved in the execution of the protocol. Alternative and equivalently, we could consider a definition of security that enumerates over a subclass of all the PT environments that *honestly* sample the parameters pp and send them as part of the inputs to all the parties.

2.3 Computational Assumptions

2.3.1 The Discrete Logarithm Assumption

Let $\mathsf{GroupGen}$ be some PPT algorithm that on input 1^λ returns a description $\mathsf{pp}_\mathbb{G}$ of a cyclic group $\mathbb{G}$: this means that every element in $\mathbb{G}$ can be written as g^x for some generator $g \in \mathbb{G}$ and exponent $x \in \mathbb{F}$ (cf. Section 2.1.1). Informally, the discrete logarithm assumption over $\mathbb{G}$ states that it is computationally infeasible, given a random group element $h \in \mathbb{G}$, to find an integer x such that $h = g^x$.

Notably, the discrete logarithm assumption does not hold against quantum polynomial-time adversaries due to Shor's algorithm [Sho97a]. However, no classical algorithm is known to compute (in generic groups[3]) discrete logarithms better than the Pollard's rho algorithm [Pol78].

Lemma 2.3.1 (Discrete Log Reduction, [GT21])**.** *For every EPT adversary $\mathcal{A}$, there exists an EPT adversary $\mathcal{B}$, nearly as efficient as $\mathcal{A}$, such that:*

$$\Pr\left[\prod_{i=1}^n g_i^{a_i} = 1 \wedge (a_1, \ldots, a_n) \neq \vec{0} \;\middle|\; (a_1, \ldots, a_n) \leftarrow \mathcal{A}(g_1, \ldots, g_n)\right] \leq \mathbf{Adv}^{\mathsf{DL}}_{\mathbb{G}}(\mathcal{B}) + \frac{1}{|\mathbb{F}|}$$

where $g, g_1, \ldots, g_n$ are random generators of $\mathbb{G}$, and we define the advantage of $\mathcal{B}$ as $\mathbf{Adv}^{\mathsf{DL}}_{\mathbb{G}}(\mathcal{B}) \coloneqq \Pr[g^x = h \mid h \leftarrow\!\!\$\ \mathbb{G}; x \leftarrow \mathcal{B}(g, h)]$.

2.3.2 Assumptions on Bilinear Groups

Definition 2.3.1 (q-strong Diffie-Hellman Assumption, [BB08])**.** *We say that the q-SDH Assumption holds in $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ if for any PPT algorithm $\mathcal{A}$ and any $q \in \mathsf{poly}(\lambda)$:*

$$\Pr\left[\mathcal{A}(\left[1, s, s^2, \ldots s^q\right]_1, [1, s]_2) = (c, \left[(s+c)^{-1}\right]_1)\right] \in \mathsf{negl}(\lambda)$$

where the probability is taken over the random choice of s and the random coins of $\mathcal{A}$.

Definition 2.3.2 (d-Power Discrete Logarithm [Lip12])**.** *Given a degree bound $d \in \mathbb{N}$, the d-Power Discrete Logarithm (d-DL) assumption holds for a bilinear group generator $\mathsf{GroupGen}$ if for every PPT adversary $\mathcal{A}$ that receives as input $(\left[1, \ldots, s^d\right]_1, \left[1, \ldots, s^d\right]_2)$, and outputs the value s', the probability that $s = s'$ is negligible. We also use DL as a shortcut for 1-DL.*

2.3.2.1 Distributions

We recall the notion of "witness sampleable" distributions, as defined by Jutla and Roy [JR13]. Informally, a distribution $\mathcal{D}$ with support in $\mathbb{G}_1$ is witness sampleable if it is possible to sample "at the exponent" following the same distribution of $\mathcal{D}$.

[3] More efficient algorithms can exist for specific groups, thus exploiting the structure of the group

Definition 2.3.3 (Witness Sampleability, [JR13]). *A distribution $\mathcal{D}$ is witness sampleable if there exists a PPT algorithm $\tilde{\mathcal{D}}$ s.t. for any $\mathsf{pp}_{\mathbb{G}}$, the random variables $\vec{A} \leftarrow\!\!\$\, \mathcal{D}(\mathsf{pp}_{\mathbb{G}})$ and $\left[\vec{\tilde{A}}\right]_1$, where $\vec{\tilde{A}} \leftarrow\!\!\$\, \tilde{\mathcal{D}}(\mathsf{pp}_{\mathbb{G}})$, are equivalently distributed.*

We now give a definition of matrix distributions.

Definition 2.3.4. *$\mathcal{D}_{n,d}$ is a* matrix distribution *if outputs (in probabilistic polynomial time, with overwhelming probability) matrices in $\mathbb{Z}_q^{n\times d}$.*

We introduce the Matrix Decisional Diffie-Hellman Assumption, introduced by Escala *et al.* [EHK+13].

Definition 2.3.5 (MDDH Assumption in $\mathbb{G}_1$, [EHK+13]). *The $\mathcal{D}_{n,d}$-MDDH assumption holds if for all non-uniform PPT adversaries $\mathcal{A}$,*

$$\left|\Pr\left[\mathcal{A}(\mathcal{G}, [\vec{A}]_1, [\vec{A}\vec{w}]_1) = 1\right] - \Pr\left[\mathcal{A}(\mathcal{G}, [\vec{A}]_1, [\vec{z}]_1) = 1\right]\right| \in \mathsf{negl}(\lambda),$$

where the probability is taken over $\mathcal{G} = (q, \mathbb{G}_1, \mathbb{G}_T, e, \mathcal{P}_1) \leftarrow \mathsf{GroupGen}(1^\lambda)$, $\vec{A} \leftarrow \mathcal{D}_{n,d}, \vec{w} \leftarrow \mathbb{Z}_q^d, [\vec{z}]_1 \leftarrow \mathbb{G}_1^n$ and the coin tosses of adversary $\mathcal{A}$.

For $Q \in \mathbb{N}$, $\vec{W} \leftarrow\!\!\$\, \mathbb{Z}_q^{d\times Q}$ and $\vec{U} \leftarrow\!\!\$\, \mathbb{Z}_q^{n\times Q}$, we consider the Q-fold $\mathcal{D}_{n,d}$-MDDH assumption, which states that distinguishing tuples of the form $([\vec{A}]_1, [A\vec{W}]_1)$ from $([\vec{A}]_1, [\vec{U}]_1)$ is hard. That is, a challenge for the Q-fold $\mathcal{D}_{n,d}$-MDDH assumption consists of Q independent challenges of the $\mathcal{D}_{n,d}$-MDDH Assumption (with the same $\vec{A}$ but different randomness $\vec{w}$). In [EHK+13] it is shown that the two problems are equivalent, where the reduction loses at most a factor $n - d$.

Lemma 2.3.2 (Random self-reducibility of $\mathcal{D}_{n,d}$-MDDH, [EHK+13]). *Let $n, d, Q \in \mathbb{N}$ with $n > d$ and $Q > n - d$. For any PPT adversary $\mathcal{A}$, there exists an adversary $\mathcal{B}$ such that $T(B) \approx T(A) + Q \cdot \mathsf{poly}(\lambda)$, with $\mathsf{poly}(\lambda)$ independent of $T(A)$, and*

$$\mathbf{Adv}^{Q-MDDH}_{\mathbb{G}_1,\mathcal{D}_{n,d},\mathcal{A}}(\lambda) \leq (n-d) \cdot \mathbf{Adv}^{mddh}_{\mathbb{G}_1,\mathcal{D}_{n,d},\mathcal{B}}(\lambda) + \frac{1}{q-1}$$

where, given $\vec{A} \leftarrow\!\!\$\, \mathcal{U}_{n,d}$, $\vec{W} \leftarrow\!\!\$\, \mathbb{Z}_q^{k\times Q}$ and $\vec{U} \leftarrow\!\!\$\, \mathbb{Z}_q^{n\times Q}$:

$$\mathbf{Adv}^{Q-MDDH}_{\mathbb{G}_1,\mathcal{D}_{n,d},\mathcal{A}}(\lambda) := |\Pr\left[\mathcal{A}(\mathcal{G}, [\vec{A}]_1, [A\vec{W}]_1) = 1\right] - \Pr\left[\mathcal{A}(\mathcal{G}, [\vec{A}]_1, [\vec{U}]_1) = 1\right]|,$$

Corollary 2.3.1. *Let $n, d, d' \in \mathbb{N}$ with $n > d$ and $d' \geq d$. For any PPT adversary $\mathcal{A}$, there exists an adversary $\mathcal{B}$ such that $T(B) \approx T(A) + \mathsf{poly}(\lambda)$, with $\mathsf{poly}(\lambda)$ independent of $T(A)$, and*

$$\mathbf{Adv}^{mddh}_{\mathbb{G}_1,\mathcal{D}_{n,d'},\mathcal{A}}(\lambda) = \mathbf{Adv}^{mddh}_{\mathbb{G}_1,\mathcal{D}_{n,d},\mathcal{B}}(\lambda).$$

We state a tighter random-self reducibility property for case of the uniform distribution $\mathcal{U}$.

Lemma 2.3.3 (Random self-reducibility of $\mathcal{U}_{n,d}$-MDDH, [EHK+13]). *Let $n, d, Q \in \mathbb{N}$ with $n > d$ and $Q > n - d$. For any PPT adversary $\mathcal{A}$, there exists an adversary $\mathcal{B}$ such that $T(B) \approx T(A) + Q \cdot \mathsf{poly}(\lambda)$, with $\mathsf{poly}(\lambda)$ independent of $T(A)$, and*

$$\mathbf{Adv}^{Q-MDDH}_{\mathbb{G}_1,\mathcal{U}_{n,d},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{mddh}_{\mathbb{G}_1,\mathcal{U}_{n,d},\mathcal{B}}(\lambda) + \frac{1}{q-1}$$

Lemma 2.3.4 ($\mathcal{D}_{n,d}$-MDDH $\Rightarrow$ $\mathcal{U}_{n,d}$-MDDH, [EHK+13]). *Let $\mathcal{D}_{n,d}$ be a matrix distribution. For any adversary $\mathcal{A}$ on the $\mathcal{U}_{n,d}$-distribution, there exists an adversary $\mathcal{B}$ on the $\mathcal{D}_{n,d}$-assumption such that $T(\mathcal{B}) \approx T(\mathcal{A})$ and $\mathbf{Adv}^{mddh}_{\mathbb{G}_1,\mathcal{U}_{n,d},\mathcal{A}}(\lambda) = \mathbf{Adv}^{mddh}_{\mathbb{G}_1,\mathcal{D}_{n,d},\mathcal{B}}(\lambda)$.*

2.4 Cryptographic primitives

2.4.1 Re-randomizable PKE

A re-randomizable PKE (Rand-PKE) scheme PKE is a tuple of five algorithms:

$\mathsf{Setup}(1^\lambda)$: upon input the security parameter 1^λ produces parameters pp, which include the description of the message and ciphertext space $\mathcal{M}$, $\mathcal{C}$.

$\mathsf{KGen}(\mathsf{pp})$: upon input the parameters pp, outputs a key pair $(\mathsf{pk}, \mathsf{sk})$.

$\mathsf{Enc}(\mathsf{pk}, \mathtt{msg})$: upon inputs a public key pk and a message $\mathtt{msg} \in \mathcal{M}$, outputs a ciphertext $\mathtt{C} \in \mathcal{C}$.

$\mathsf{Dec}(\mathsf{pk}, \mathsf{sk}, \mathtt{C})$: upon inputs a secret key sk and a ciphertext $\mathtt{C}$, outputs a message $\mathtt{msg} \in \mathcal{M}$ or an error symbol $\perp$.

$\mathsf{Rand}(\mathsf{pk}, \mathtt{C})$: upon inputs a public key pk and a ciphertext $\mathtt{C}$, outputs another ciphertext $\mathtt{C}'$.

We move now to the definition of perfect re-randomizability as defined in [FFHR19]. Roughly speaking, a PKE is re-randomizable if there exists a procedure that creates fresh and unlinkable ciphertexts from an old ciphertext.

Definition 2.4.1 (Perfect Re-randomizability, [FFHR19])**.** *We say that* PKE *is perfectly re-randomizable (Re-Rand, for short) if the following three conditions are met:*

(Indistinguishability) *For any* $\lambda \in \mathbb{N}$*, any* $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$*, any* $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{pp}, 1^\lambda)$*, for any* $\mathtt{msg} \in \mathcal{M}$ *and any* $\mathtt{C} \in \mathsf{Enc}(\mathsf{pk}, \mathtt{msg})$ *the following two distributions are identical*

$$\mathtt{C}_0 \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{pk}, \mathtt{msg}) \text{ and } \mathtt{C}_1 \leftarrow\!\!\$\ \mathsf{Rand}(\mathsf{pk}, \mathtt{C});$$

(Correctness) *For any* $\lambda \in \mathbb{N}$*, any* $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$*, any* $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{pp}, 1^\lambda)$*, for any (possibly malicious) ciphertext* $\mathtt{C}$ *and every* $\mathtt{C}' \leftarrow\!\!\$\ \mathsf{Rand}(\mathsf{pk}, \mathtt{C})$ *it holds*

$$\mathsf{Dec}(\mathsf{sk}, \mathtt{C}') = \mathsf{Dec}(\mathsf{sk}, \mathtt{C}).$$

(Tightness of Decryption) *For any (possibly unbounded) adversary* $\mathcal{A}$ *and any sequence of parameters* $\{\mathsf{pp}_\lambda \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)\}_{\lambda \in \mathbb{N}}$ *the following holds:*

$$\Pr\left[\exists \mathtt{msg} : \mathtt{C} \notin \mathsf{Enc}(\mathsf{pk}, \mathtt{msg}) \wedge \mathsf{Dec}(\mathsf{sk}, \mathtt{C}) = \mathtt{msg} \neq \perp : \begin{array}{c}(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{pp}_\lambda) \\ \mathtt{C} \leftarrow\!\!\$\ \mathcal{A}(\mathsf{pk})\end{array}\right] \in \mathsf{negl}(\lambda).$$

Finally, we define the notion of public verifiability for PKE schemes.

Definition 2.4.2 (Public Verifiability)**.** $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Rand})$ *is a public key scheme with publicly verifiable ciphertexts if there is a deterministic algorithm* Verify *which, on input* $(\mathsf{pk}, \mathtt{C})$ *outputs an error symbol* $\perp$ *whenever* $\mathsf{Dec}(\mathsf{pk}, \mathsf{sk}, \mathtt{C}) = \perp$*, else it outputs valid.*

2.4.2 Commitment Schemes

The notion of commitment is one of the simplest and most used in cryptography. Informally, it is a scheme that allows a party (the committer) to create a commitment to a secret value so that it is possible to later open the commitment and reveal the value in a verifiable manner. Ideally, a commitment should *hide* the secret value committed and, at the same time, it should be *binding*, in the sense that a commitment should only be opened to a single value. We proceed by giving a formal definition.

Definition 2.4.3 (Commitment scheme)**.** *A commitment scheme with message space $\mathcal{M}$ (and group parameters* $\mathsf{GroupGen}$*) is a tuple of algorithms* $\mathsf{CS} = (\mathsf{KGen}, \mathsf{Com}, \mathsf{VerCom})$ *that works as follows:*

$\mathsf{KGen}(\mathsf{pp}_{\mathbb{G}}) \to \mathsf{ck}$ *takes as input group parameters* $\mathsf{pp}_{\mathbb{G}} \leftarrow\!\!\$\ \mathsf{GroupGen}(1^\lambda)$ *and outputs a commitment key* ck*.*

$\mathsf{Com}(\mathsf{ck}, m) \to (c, o)$ *takes the commitment key* ck*, and a message $m \in \mathcal{M}$, and outputs a commitment c and an opening o.*

$\mathsf{VerCom}(\mathsf{ck}, c, m, o) \to b$ *takes as input the commitment key* ck*, a commitment c, a message $m \in \mathcal{M}$ and an opening o, and it accepts ($b = 1$) or rejects ($b = 0$).*

A commitment scheme CS satisfies the following properties.

Correctness For any $\lambda \in \mathbb{N}$, any commitment key $\mathsf{ck} \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$, any message $m \in \mathcal{M}$, and for any honestly generated commitment-opening $(c, o) \leftarrow\!\!\$\ \mathsf{Com}(\mathsf{ck}, m)$, we have that $\mathsf{VerCom}(\mathsf{ck}, c, m, o) = 1$

Binding For every (non-uniform) efficient adversary $\mathcal{A}$:

$$\Pr\left[\begin{array}{c}\mathsf{VerCom}(\mathsf{ck}, c, m, o) = 1 \\ \wedge \\ \mathsf{VerCom}(\mathsf{ck}, c, m', o') = 1\end{array} : \begin{array}{r}\mathsf{ck} \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda) \\ (c, m, o, m', o') \leftarrow \mathcal{A}(\mathsf{ck})\end{array}\right] \in \mathsf{negl}(\lambda)$$

Hiding: $\forall m, m'$, $\forall \mathsf{ck}$:

$$\{c : (c, o) \leftarrow \mathsf{Com}(\mathsf{ck}, m)\} \approx \{c' : (c', o') \leftarrow \mathsf{Com}(\mathsf{ck}, m')\}$$

In some cases, we relax the hiding property, and we require the following instead.

Trapdoor-Hiding There exist three algorithms $\mathcal{S}_{\mathsf{ck}}, \mathsf{TdCom}, \mathsf{TdOpen}$:

$\mathcal{S}_{\mathsf{ck}}(1^\lambda) \to (\mathsf{ck}, \mathsf{td})$ on input the security parameter outputs a commitment key ck and a trapdoor td,

$\mathsf{TdCom}(\mathsf{ck}) \to (c, st)$ on input td outputs a commitment c and a trapdoor opening st,

$\mathsf{TdOpen}(\mathsf{ck}, st, c, m) \to o$ on input ck, trapdoor opening st, a commitment c and a message $m \in \mathcal{M}$, outputs an opening o,

such that (i) the distribution of the commitment key returned by $\mathcal{S}_{\mathsf{ck}}$ is perfectly/statistically close to the one of the key returned by KGen; (ii) for any $m \in \mathcal{M}$, $(c, o) \approx (c', o')$ where $(c, o) \leftarrow \mathsf{Com}(\mathsf{ck}, m)$, $(c', st) \leftarrow \mathsf{TdCom}(\mathsf{td})$ and $o' \leftarrow \mathsf{TdOpen}(\mathsf{td}, st, c', m)$.

2.4.2.1 Polynomial Commitments

A polynomial commitment scheme is essentially a commitment scheme that is equipped with a protocol (either interactive or non-interactive) that allows the prover to convince the verifier that the object he committed to is a polynomial (whose degree is upper-bounded by a known parameter d) that evaluates to some value y at some point x. For a concrete example, we refer to the Hyrax commitment scheme that we describe in Fig. 5.8 using this syntax.

However, there is a different perspective we can take.

Definition 2.4.4 (Polynomial Commitment)**.** *A polynomial commitment* PC *is a commitment scheme (Definition 2.4.3) whose message space is* $\mathbb{F}_{\leq d}[X]$*, the set of low degree polynomials over a finite field* $\mathbb{F}$ *with degree bound* $d \in \mathbb{N}$*. The key generation algorithm might take as additional input the degree* d*.*

A polynomial commitment scheme, then, is just a commitment scheme (for a specific choice of the message space). If the prover wishes to convince the verifier about the evaluation of a polynomial f he committed to, say on point x and evaluation value y, he can use a CP-SNARK for the relation $\mathcal{R}_{\mathtt{evl}}((x,y),f) := f(x) = y$. We prefer adopting this perspective because it is more flexible.

2.4.2.2 Commitment Instantiations

In this section we introduce some of the commitment schemes that we will use throughout the manuscript.

One of the most well-known commitment schemes is the Pedersen commitment scheme. It is perfectly hiding, and its binding property relies on the discrete logarithm assumption.

Construction 2.4.1 (Pedersen)**.** $\mathtt{Pedersen}$ *is a commitment scheme for the message space* $\mathbb{F}^n$*, for some* $n \in \mathbb{N}$*, defined over a cyclic group* $\mathbb{G}$ *equipped with a group generator algorithm* $\mathsf{GroupGen}$*, and consists of the following algorithms:*

$\mathsf{KGen}(1^\lambda)$ *on input the security parameter* 1^λ*, outputs* $n+1$ *random generators* $g_1, \ldots, g_n, h$ *of* $\mathbb{G}$

$\mathsf{Com}(\mathsf{ck}, \vec{m}; \omega)$ *on input the commitment key* $\mathsf{ck} := (g_1, \ldots, g_n, h)$*, and a vector of messages* $\vec{m} \in \mathbb{F}^n$*, outputs a commitment* $c := \prod_{i\in[n]} g_i^m h^\omega$ *and the opening* ω

$\mathsf{VerCom}(\mathsf{ck}, c, \vec{m}, \omega)$ *outputs 1 if* $c = \prod_{i\in[n]} g_i^m h^\omega$*, and 0 otherwise*

We now recap the polynomial commitment scheme of Kate, Zaverucha and Goldberg [KZG10] (KZG, shortly), highlighting in purple the parts related to hiding.

Construction 2.4.2 (KZG)**.** *KZG is a Polynomial Commitment scheme (see Definition 2.4.4) defined over bilinear groups* $\mathbb{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$*, that consists of the following algorithms:*

$\mathsf{KGen}(1^\lambda, d)$ *on input the security parameter* 1^λ*, and a maximum degree bound* $d \in \mathbb{N}$*, outputs* $\mathsf{ck} := (([s^j]_1)_{j\in[0,d]}, ([\alpha s^j]_1)_{j\in[0,d]}, [1,s]_2)$*, for a random secret* $s, \alpha \leftarrow\!\!\$\ \mathbb{F}_q$*.*

$\mathsf{Com}(\mathsf{ck}, f(X), r(X))$ *on input* ck*, a polynomial* $f(X)$*, and a masking polynomial* $r(X)$*, outputs a commitment* $c := [f(s)+\alpha r(s)]_1$*.*

$\mathsf{VerCom}(\mathsf{ck}, c, f(X))$ *outputs 1 if* $c = [f(s)+\alpha r(s)]_1$.

The above scheme is (standard) binding under the d-DL assumption (see Groth [Gro10a]), in fact, given two polynomials f and f' that evaluate to the same value on the secret point s, we can find s among the roots of the (non-zero) polynomial $f - f'$.

2.4.3 Cryptographic proofs and arguments

NP relations. Following Groth et al. [GKM+18], an (indexed) NP relation $\mathcal{R}$ is a set of tuples $(\mathsf{pp}, \mathbb{x}, \mathbb{w})$ decided by a PT algorithm. Here pp are system-wide parameters,[4] $\mathbb{x}$ is the public input (or instance), and $\mathbb{w}$ is the private input (or witness). We interchangeably represent a relation $\mathcal{R}$ either as an algorithm with boolean output or as a set, thus $\mathcal{R}(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \iff (\mathsf{pp}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$. Moreover, when clear from the context, we omit the parameters and simply write $\mathcal{R}(\mathbb{x}, \mathbb{w})$.

2.4.3.1 NIZKs

In this section, we define the basic syntax and properties for a Non-Interactive Zero-Knowledge Argument of Knowledge (NIZK).

A NIZK for a relation $\mathcal{R}$ (and group generator $\mathsf{GroupGen}$) is a tuple of algorithms $\Pi = (\mathsf{KGen}, \mathsf{Prove}, \mathsf{Verify})$ that work as described below.

- $\mathsf{KGen}(\mathsf{pp}_{\mathbb{G}}) \to \mathsf{srs}$ is a probabilistic algorithm that takes as input the group parameters $\mathsf{pp}_{\mathbb{G}} \leftarrow\!\$\ \mathsf{GroupGen}(1^\lambda)$ and outputs $\mathsf{srs} := (\mathsf{ek}, \mathsf{vk}, \mathsf{pp})$, where ek is the evaluation key, vk is the verification key, and pp are the parameters for the relation $\mathcal{R}$.
- $\mathsf{Prove}(\mathsf{ek}, \mathbb{x}, \mathbb{w}) \to \pi$ takes an evaluation key ek, a statement $\mathbb{x}$, and a witness $\mathbb{w}$ such that $\mathcal{R}(\mathsf{pp}, \mathbb{x}, \mathbb{w})$ holds, and returns a proof π.
- $\mathsf{Verify}(\mathsf{vk}, \mathbb{x}, \pi) \to b$ takes a verification key, a statement $\mathbb{x}$, and either accepts ($b = 1$) or rejects ($b = 0$) the proof π.

Basic notions for a NIZK are completeness, knowledge-soundness and zero-knowledge.

Completeness: For all $\mathsf{srs} \leftarrow\!\$\ \mathsf{KGen}(\mathsf{pp}_{\mathbb{G}})$ and all $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$, we have that

$$\mathsf{Verify}(\mathsf{vk}, \mathbb{x}, \mathsf{Prove}(\mathsf{ek}, \mathbb{x}, \mathbb{w})) = 1$$

Knowledge-soundness: For every PPT adversary $\mathcal{A}$, there exists a PPT extractor $\mathcal{E}_{\mathcal{A}}$ such that:

$$\Pr[(\mathbb{x}, \mathbb{w}) \notin \mathcal{R} \wedge \mathsf{Verify}(\mathsf{vk}, \mathbb{x}, \pi) = 1 \mid (\mathbb{x}, \pi) \leftarrow \mathcal{A}(\mathsf{srs}); \mathbb{w} \leftarrow \mathcal{E}_{\mathcal{A}}(\mathsf{srs})] \in \mathsf{negl}(\lambda)$$

Zero-Knowledge. We explicitly define zero-knowledge in the SRS and RO model.[5] The zero-knowledge simulator $\mathcal{S}$ of a NIZK is a stateful PPT algorithm that can operate in three modes:

[4] For example, pp could be the description of a bilinear group or additionally contain a commitment key for a commitment scheme or a common reference string.

[5] In Chapter 5 we will introduce a different notion of trapdoor-less zero-knowledge in which, however, the simulator has the possibility to reprogram the random oracle.

- $(\mathsf{srs}, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}(0, \mathsf{pp}_{\mathbb{G}})$ takes care of generating the parameters and the simulation trapdoor (if necessary)
- $(\pi, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}(1, \mathsf{st}_{\mathcal{S}}, \mathbb{x})$ simulates the proof for a statement $\mathbb{x}$
- $(a, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}(2, \mathsf{st}_{\mathcal{S}}, s)$ takes care of answering random oracle queries

The state $\mathsf{st}_{\mathcal{S}}$ is updated after each operation.

Definition 2.4.5 (Zero-Knowledge)**.** *A NIZK* NIZK *is (perfect) zero-knowledge if there exists a PPT simulator* $\mathcal{S}$ *such that for all adversaries* $\mathcal{A}$*:*

$$\Pr\left[\begin{array}{l}\mathsf{pp}_{\mathbb{G}} \leftarrow \mathsf{GroupGen}(1^\lambda)\\ \mathsf{srs} \leftarrow \mathsf{KGen}(\mathsf{pp}_{\mathbb{G}})\\ \mathcal{A}^{\mathsf{Prove}(\mathsf{ek},\cdot,\cdot)}(\mathsf{srs}) = 1\end{array}\right] \approx \Pr\left[\begin{array}{l}\mathsf{pp}_{\mathbb{G}} \leftarrow \mathsf{GroupGen}(1^\lambda)\\ (\mathsf{srs}, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}(0, \mathsf{pp}_{\mathbb{G}})\\ \mathcal{A}^{\mathcal{S}'(\cdot,\cdot)}(\mathsf{srs}) = 1\end{array}\right]$$

where $\mathcal{S}'$ *is an oracle that on input a pair* $(\mathbb{x}, \mathbb{w})$ *first checks* $(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$ *where* pp *is part of* srs *and then returns the first output of* $\mathcal{S}(1, \mathsf{st}_{\mathcal{S}}, \mathbb{x})$*.*

Rewinding. A common technique used to prove the security of a NIZK is the *rewinding* approach. The simulator/extractor runs the code of the prover, feeding it with the verifier inputs it requires, and then has the possibility to *rewind* it to some previous state to feed it with different inputs. This technique is useful when the extractor needs to get several outputs of the prover with respect to different verifier inputs: roughly speaking, this allows to extract the (possibly large and masked) witness from a (possibly succinct) proof.

In Chapter 5 we will make use of rewinding, and we will define knowledge-soundness in a setting in which the extractor can explicitly rewind the prover.

2.4.3.2 zkSNARKs

Zero-knowledge succinct argument of knowledge (zkSNARKs) are NIZKs that are *succinct* as defined below.

Definition 2.4.6 (Succinctness)**.** *A NIZK* Π *is said succinct if the running time of* Verify *is* $\mathsf{poly}(\lambda + |\mathbb{x}| + \log|\mathbb{w}|)$ *and the proof size is* $\mathsf{poly}(\lambda + \log|\mathbb{w}|)$*.*

CP-SNARKs. Commit-and-Prove SNARKs (CP-SNARKs) are zkSNARKs in which the relations verify predicates over commitments (see Campanelli, Fiore and Querol [CFQ19]).

Briefly speaking, we refer to a CP-SNARK for a relation $\mathcal{R}$ and a commitment scheme CS as a tuple of algorithms $\mathsf{CP} = (\mathsf{KGen}, \mathsf{Prove}, \mathsf{Verify})$ where:

- $\mathsf{KGen}(\mathsf{ck}) \to \mathsf{srs}$ is a probabilistic (or deterministic) algorithm that takes as input a commitment key ck for CS and outputs $\mathsf{srs} := (\mathsf{ek}, \mathsf{vk}, \mathsf{pp})$, where ek is the evaluation key, vk is the verification key, and pp are the parameters for the relation $\mathcal{R}$ (which include the commitment key ck).

Moreover, if we consider the key generation algorithm KGen' that upon group parameters $\mathsf{pp}_{\mathbb{G}}$ first runs $\mathsf{ck} \leftarrow\!\!\$\ \mathsf{CS.KGen}(\mathsf{pp}_{\mathbb{G}})$, runs $\mathsf{srs} \leftarrow\!\!\$\ \mathsf{CP.KGen}(\mathsf{ck})$ and outputs srs; then the tuple $(\mathsf{KGen}', \mathsf{Prove}, \mathsf{Verify})$ defines a SNARK.

2.4.3.3 Compilation of zkSNARKs

A common approach to design zkSNARKs is to first construct an information-theoretic protocol that achieves the desired functionality in an idealized model and then remove the idealized component by compiling it into a zkSNARK via the use of a computationally-secure primitive [Ish19, Ish20]. The most popular instantiation of this approach uses a Polynomial Interactive Oracle Proof (PIOP) [GWC19, BFS20, CHM+20, Sze20, CFF+21] for the information-theoretic part, and a polynomial commitment [KZG10] for the computational one.

We give a formal definition of (a specific class of) PIOPs in the next section, but first, we give some background on the compilation strategy.

In a PIOP, the prover uses one oracle *to commit* to polynomials while the verifier calls a second oracle *to query* the committed polynomials. In the compiled PIOP, instead, the prover commits to polynomials with a polynomial commitment, and then computes the results of verifier's queries and uses the *evaluation opening* to vouch for their correctness. Finally, to remove interaction the compiler employs the Fiat-Shamir transformation to obtain the zkSNARK. The details of the (kind of) verifier's queries often diverge in different implementations of this paradigm. Arguably, the simplest form of queries is the evaluation of polynomials, namely, queries checking that a committed polynomial p at evaluation point x evaluates to $y = p(x)$; this is the model used in [CHM+20, BFS20]. Other PIOP variants [GWC19, CFF+21] consider more general queries that state the validity of polynomial equations over (a subset) of the committed polynomials.

2.4.3.4 Polynomial Holographic Interactive Oracle Proofs

In this section, we give a formal definition of PIOPs. We focus on the class of Polynomial (Holographic) IOPs defined by [CFF+21] as a generalization of [BFS20]. PIOPs can flexibly capture under the same hat all the most recent protocols based on the notions of [BFS20], AHP [CHM+20], and ILDP [GWC19].

The oracle of the prover commits to low-degree polynomials over a finite field while the queries of the verifier check polynomial equations over these polynomials. These polynomial equations can depend on additional field elements sent by the prover and/or the verifier during the execution of the protocol. Slightly more in detail, the verifier can query an oracle polynomial $p(X)$ (or multiple polynomials simultaneously) by specifying polynomials G and v to test equations of the form $G(X, p(v(X))) \equiv 0$. Therefore, to be compiled, PIOPs need a commit-and-prove SNARK (CP-SNARK) for proving the validity of such equations concerning the committed polynomials. Notably, one can easily build this CP-SNARK from a CP-SNARK for polynomial evaluations (e.g., KZG) by testing the equations on a random point chosen by the random oracle.

Definition 2.4.7 (Polynomial Holographic IOP)**.** *Let $\mathcal{F}$ be a family of finite fields and let $\mathcal{R}$ be an indexed relation. A (public-coin non-adaptive) Polynomial Holographic IOP over $\mathcal{F}$ for $\mathcal{R}$ is a tuple $\mathsf{PIOP} = (r, n, m, D, n_e, \mathsf{I}, \mathsf{P}, \mathsf{V})$ where $r, n, m, D, n_e \colon \{0,1\}^* \to \mathbb{N}$ are polynomial-time computable functions, and $\mathsf{I}, \mathsf{P}, \mathsf{V}$ are three algorithms for the encoder, prover and verifier respectively, that work as follows.*

Offline phase: *The indexer $\mathsf{I}(\mathbb{F}, \mathtt{i})$ is executed on input a field $\mathbb{F} \in \mathcal{F}$ and a relation description $\mathtt{i}$, and it returns $n(0)$ polynomials $\{p_{0,j}\}_{j \in [n(0)]}$ encoding the relation $\mathtt{i}$.*

Online phase: *The prover* $\mathsf{P}(\mathbb{F}, \mathbb{i}, \mathbb{x}, \mathbb{w})$ *and the verifier* $\mathsf{V}^{\mathsf{I}(\mathbb{F},\mathbb{i})}(\mathbb{F}, \mathbb{x})$ *are executed for* $r(|\mathbb{i}|)$ *rounds; the prover has a tuple* $(\mathbb{F}, \mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$ *and the verifier has an instance* $\mathbb{x}$ *and oracle access to the polynomials encoding* $\mathbb{i}$*.*

In the i*-th round,* P *sends* $m(i)$ *messages* $\{\pi_{i,j} \in \mathbb{F}\}_{j\in[m(i)]}$*, and* $n(i)$ *oracle polynomials* $\{p_{i,j} \in \mathbb{F}[X]\}_{j\in[n(i)]}$ *of degree at most* $D \coloneqq D(|\mathbb{i}|)$*, while* V *replies (except for the last round) with a uniformly random message* $\rho_i \in \mathbb{F}$*.*

Decision phase: *After the* $r \coloneqq r(|\mathbb{i}|)$*-th round, let* $n_e \coloneqq n_e(|\mathbb{i}|)$*, the verifier* $\mathcal{V}(\mathbb{F}, \mathbb{x}, \vec{\rho})$*, on input the description of the field* $\mathbb{F}$*, the input* $\mathbb{x}$ *and all the random messages of the verifier* $\vec{\rho} \coloneqq (\rho_1, \ldots, \rho_{r-1})$*, outputs tuples* $(G^{(k)}, v_1^{(k)}, \ldots, v_n^{(k)})_{k\in[n_e]}$ *which define the following algebraic checks. Let* $n \coloneqq \sum_{k=0}^{r} n(k)$*, let* $m \coloneqq \sum_{k=1}^{r(|\mathbb{i}|)} m(k)$*, and denote by* $(p_1, ..., p_n)$ *all the oracle polynomials (including the* $n(0)$ *ones from the encoder) and by* $(\pi_1, ..., \pi_m)$ *all the messages sent by the prover. For any* $k \in [n_e], j \in [n]$ *we have* $G^{(k)} \in \mathbb{F}[X, X_1, ..., X_n, Y_1, ..., Y_m]$ *and* $v_j^{(k)} \in \mathbb{F}[X]$*. A tuple* $(G^{(k)}, v_1^{(k)}, ..., v_n^{(k)})$ *is satisfied if and only if* $F^{(k)}(X) \equiv 0$ *where:*

$$F^{(k)}(X) \coloneqq G^{(k)}(X, \{p_i(v_i^{(k)}(X))\}_{i\in[n]}, \{\pi_i\}_{i\in[m]}) \tag{2.2}$$

The verifier accepts if and only if all the checks are satisfied.

Chapter 3

Non-malleable Polynomial commitments and zkSNARKs

This chapter is extracted from "From Polynomial IOP and Commitments to Non-malleable zkSNARKs", published in TCC 2023.

3.1 Introduction

The design of modern zkSNARKs follows the common cryptographic approach of starting with protocols that achieve information-theoretic security in idealized models and then compiling them into efficient protocols by employing a smaller computationally secure primitive.

As discussed in Section 2.4.3.3, in the world of SNARKs, the corresponding concepts are (polynomial) interactive oracle proofs F-IOP [BFS20, CFF+21, CHM+20, GWC19, Sze20] and (polynomial) functional commitments F-COM [BDFG20, KZG10, Lee21]. An F-IOP employs two (idealized) oracles that share their state: the prover calls the first oracle *to commit* to functions $f \in F$ and the verifier calls the second *to query* the committed functions. Concretely, the F-IOP to SNARK compiler uses F-COM to replace oracles with commitments, opening proofs, and query proofs. As this only removes reliance on idealized function oracles but not interaction, the compiler additionally employs the usual Fiat-Shamir transformation for public-coin protocols to obtain the final zkSNARK. The benefits of this compilation paradigm are modularity and separation of concerns: once the compiler is proven, a line of research can address the problem of improving F-IOPs while another research line can tackle the problem of realizing F-COM schemes (e.g., with better efficiency, from different assumptions, etc.): this approach has been successfully adopted to construct several recent zkSNARKs. All this recent work, though, only shows that schemes obtained via this paradigm are knowledge-sound.

3.1.1 Our contributions

We study the simulation extractability of a broad class of zkSNARKs built through this "natural" compilation approach. In particular, our primary goal includes showing that not only *existing* zkSNARKs but also any future zkSNARK following this, by now standard, construction framework, provide simulation extractability. This goal has a twofold motivation. On the theoretical side, we are interested in understanding sufficient conditions on F-COM to compile an F-IOP into a simulation-extractable zkSNARK. On the practical side,

by capturing existing compilers we can show that existing schemes that are under deployment, e.g., Plonk [GWC19], have already this strong security property.[1] For this reason, in our work we focus on the popular case of the compiler where the F-IOP is a polynomial IOP (cf. Section 2.4.3.4), i.e., the oracle functions F are low-degree polynomials, and F-COM is a polynomial commitment (see Section 2.4.2.1). Furthermore, in terms of instantiations we are interested to cover the celebrated KZG polynomial commitment scheme [KZG10] and on a polynomial IOP framework that is flexible enough to include recent constructions, e.g., [CFF+21, CHM+20, GWC19, MBKM19, RZ21]. The main contributions of our work are:

- to introduce a relaxed notion of simulation extractability for polynomial commitments;
- to prove that the KZG scheme satisfies our relaxed SE notion in the algebraic group model (AGM) and random oracle model (ROM);
- to prove that our notion is sufficient to compile a polynomial IOP into a (full-fledged) simulation-extractable zkSNARK, using the usual compilation approach.

By combining these results we obtain a simulation extractability proof for Plonk [GWC19], Basilisk [RZ21], and a slight variation of Marlin [CHM+20] and Lunar [CFF+21].

3.1.2 Organization of the chapter

In Section 3.2, we elaborate more on our results and the technical challenges that we had to overcome along the way. In Section 3.3 we introduce some of the computational assumptions that will be used to prove our results. We define the framework of policy-based simulation extractability in Section 3.4, we proceed with the analysis of the simulation extractability of KZG in Section 3.5, and we give our generic compiler for strong simulation-extractable Universal SNARKs from (simulation-extractable) polynomial commitment and PIOP in Section 3.6. The thesis of the theorem on KZG polynomial commitment and the hypothesis for the theorem of the Universal SNARKs compiler do not quite match. Indeed, they could be even considered as two independent and (almost) self-contained results. We show how to fill the gaps and connect the two results in Section 3.6.3.

3.2 A Technical Overview of Our Results

3.2.1 Simulation extractability challenges

Intuitively, the use of a simulation-extractable CP-SNARK in the above compiler should result in a simulation-extractable zkSNARK: the zero-knowledge simulator samples random commitments (relying either on hiding property of commitments, or the randomness in the committed functions p). It then simulates evaluations of p that satisfy the verification equation of the PIOP. The reduction to PIOP soundness extracts all committed polynomials from their opening proofs and the final polynomial evaluations from the evaluation proofs. However, this approach presents two major challenges:

[1] In fact as Mahak Pancholi and Akira Takahashi informed us of a flaw in the trapdoor-less zero-knowledge simulation of [GKK+22] this is arguably the first proof of simulation extractability of unmodified Plonk, i.e., Plonk with deterministic KZG commitments.

- The PIOP could be arbitrary. For example, consider a PIOP obtained by the sequential composition of two PIOP protocols for two independent statements. Very likely, the set of queries to the polynomials made by the two sub-protocols are independent and (unless the PIOP specifies it explicitly) the evaluation queries of the first sub-protocol may be chosen based on the verifier's random challenges sent *before* the second sub-protocol even starts. The simulation extractability of the zkSNARK compiled from this protocol might be affected because one could strip off the second set of evaluation proofs and replace them with those for another statement[2].

- Secondly, one needs to prove that existing, efficient, and practically deployable instantiations of polynomial commitment schemes are simulation-extractable.

3.2.2 Our solutions

To solve the first challenge, motivated by our goal to show that existing zkSNARKs are simulation-extractable and that future schemes can seamlessly achieve simulation extractability, we define a (rather minimal) constraint on the PIOP. Namely, we require that at least one of the polynomial equations involves all the oracle polynomials and that the polynomial v chosen by the verifier (see above) is not constant.[3] Fortunately, this constraint is natural and easy to meet in practice: Plonk naturally meets our constraint meanwhile all the other proof systems based on Aurora's univariate sumcheck [BCR+19] can be easily (and at negligible cost) adapted by instantiating the proof of polynomial degree through an evaluation query on all the polynomials.

For the second challenge, unfortunately, the issue is that the most obvious candidate, the efficient and widely deployed KZG polynomial commitment scheme [KZG10], is not simulation-extractable. Using bracket notation, KZG commitments are of the form $[p(s)]_1$ for a trapdoor secret s encoded in the parameters $([s^i]_1)_{i\in[0..d]}, [1, s]_2$, while evaluation proofs for an input x and output y are of the form $[\frac{p(s)-y}{s-x}]_1$. KZG is malleable, for example, given a commitment to p anyone can compute $[p(s) + \Delta]_1$ and open it using the same proof to $(x, y + \Delta)$.

Our starting point is the observation that KZG retains a form of simulation extractability for evaluations at points that are randomly chosen *after* the commitment. Fortunately, this is the situation we encounter in the Fiat-Shamir part of the PIOP-to-SNARK compiler. The commitment forms part of the first commit-and-prove part of the statement which is hashed to determine the x of the second part of the statement. Thus, the evaluation point depends on the commitment and can be considered random in the RO model.

To formalize this important relaxation, we introduce the notion of policy-based simulation extractability (Φ-SE, w.r.t. a policy parameter Φ). In the standard simulation extractability experiment, the adversary can ask the simulator to generate proofs for statements of its choice and, eventually, must produce a *new* valid proof without knowing the witness. In Φ-SE, we consider a relaxation of the SE game in which all the simulation queries of the adversary

[2] In particular, the adversary could have a simulated proof $\tilde{\pi} = (\pi, \pi')$ for $(\mathbb{x}, \mathbb{x}')$ and then could choose $\mathbb{x}''$ for which it knows a valid witness, and finally forge for $(\mathbb{x}, \mathbb{x}'')$ using (π, π''), where π'' is honestly generated. As the simulated proof π is reused, extraction fails. Notice, this attack works even when the Fiat-Shamir challenges for π'' are derived by hashing a transcript that contains π.

[3] This can be for example be implemented via a common random point chosen at the end of the protocol, on which all oracles are evaluated.

must satisfy a predicate specified in Φ; similarly, Φ can constrain the winning condition of the adversary. For this reason, we refer to Φ as the *policy*. One can see that Φ-SE is a generalization of existing SE notions such as true-simulation extractability (where the adversary can only see simulated proofs on true statements) [DHLW10], or weak simulation extractability (where the adversary only wins if it provides a proof for a new statement and, contrary to (strong) SE, loses if it provides a new proof for a statement previously asked to the simulation oracle).

Once having defined this framework, we analyze which policies Φ are strong enough to achieve simulation extractability in the compiled zkSNARK, while at the same time being weak enough for instantiation by KZG under plausible assumptions (in the AGM [FKL18] and RO). Specifically, we isolate the (simulation) extractability properties needed for the compiler and verify it for KZG in the AGM. This is the only part of our results where we need the AGM. Given the broad applications of KZG in the field of practical zkSNARKs and beyond, the characterization of its (non-)malleability is interesting in its own right. In fact, our policy highlights some malleability attacks that we discovered and that we needed to handle. Finally, for our Φ we prove that KZG is Φ-SE in the AGM and ROM. This proof turned out to be highly non-trivial and is one of the main technical contributions of our work.

Looking ahead. An advantage of our formalization of PIOP over previous proposals such as [BFS20] is that it naturally supports optimization tricks in the literature [CFF+21]. As an intermediate step of our compiler, we define a CP-SNARK for polynomial evaluations based on KZG. While we capture the important use case of batched evaluations on a common point, for the sake of simplicity, we do not capture the case of proving evaluations on *arbitrary* linear combinations of committed polynomials. We will show in Chapter 4 how to extend our compiler to support this case.

3.3 New computational assumptions

3.3.1 The Affine Matrix Diffie-Hellman Assumption

We introduce a new family of computational assumptions, named *Affine Matrix Diffie-Hellman Problem*.

Definition 3.3.1 ($\mathcal{D}_{\ell,k}$-Aff-MDH assumption)**.** *Given a matrix distribution $\mathcal{D}_{\ell,k}$, the Affine Diffie-Hellman Problem is: given $\vec{A} \in \mathbb{G}_1^{\ell \times k}$, with $\vec{A} \leftarrow\!\!\$\ \mathcal{D}_{\ell,k}$, find a nonzero vector $\vec{x} \in \mathbb{Z}_q^{\ell}$ and a vector $\vec{y} \in \mathbb{Z}_q^{k}$ such that $\left[\vec{x}^\top \vec{A}\right]_1 = [\vec{y}]_1$.*

The Aff-MDH Assumption could be seen as an extension of the Kernel-MDH Assumption introduced by Morillo, Ràfols, Villar [MRV16] since the Ker-MDH assumes $\vec{y} = \vec{0}$; however, our assumption is incomparable because we also require the adversary to give $\vec{x} \in \mathbb{Z}_q^{\ell}$ "in the exponent".

We show that the Aff-MDH Assumption, when restricted to the uniform random distribution $\mathcal{U}_{\ell,k}$, can be reduced to the discrete logarithm assumption.

Lemma 3.3.1 (DL $\Rightarrow \mathcal{U}_{\ell,k}$-Aff-MDH)**.**

Proof. The reduction samples at the exponent a uniformly random matrix $\vec{A} = (a_{i,j})_{i,j} \in \mathbb{Z}_q^{\ell \times k}$ and invokes the adversary on input $[(a_{i,j})_{i,j}]_1$. Finally, let $p_i(s)$ be the i-the row of $\vec{x}^\top \vec{A}$. The reduction computes s by factoring one of the k polynomials $p_i(s) - y_i$. □

3.3.2 The Power Polynomial in the Exponent Assumption

Definition 3.3.2 ((d, d')-Power Polynomial in the Exponent)**.** *The (d, d')-PEA Assumption holds for a bilinear group generator* GroupGen *if for every PPT adversary $\mathcal{A}$ that receives as input $([1, \ldots, s^d]_1, [1, \ldots, s^{d'}]_2)$ and outputs a polynomial $p(X)$ of degree at most $\max\{d, d'\}$, and a value y, the probability that $p(s) = y$ is negligible.*

When $d = d'$ we use the shortcut d-PEA.
Finally, we show a reduction from d-DL to d-PEA.

Lemma 3.3.2 (d-DL $\Rightarrow$ d-PEA)**.**

Proof. We can make a reduction to the assumption that computes s. The reduction invokes the adversary, gets $p(X) - y$ of degree d, and computes s by factoring the polynomial $p(s) - y$. As $p(s) - y = 0$ we are guaranteed that s is a root. □

3.4 Policy-based Simulation-Extractable NIZKs

First, we slightly modify the definition of zero-knowledge given in Definition 2.4.5 to account for the fact that we allow the simulator to take as additional input some auxiliary information aux.

Zero-Knowledge Simulators. The zero-knowledge simulator $\mathcal{S}$ of a NIZK is a stateful PPT algorithm that can operate in three modes (cf. Definition 2.4.5). Similarly to [FKMV12, GOP+22], we define the following wrappers.

Definition 3.4.1 (Wrappers for NIZK Simulator)**.** *The following algorithms are stateful and share their state $\mathsf{st} = (\mathsf{st}_{\mathcal{S}}, \mathsf{coms}, \mathcal{Q}_{\mathsf{sim}}, \mathcal{Q}_{\mathsf{RO}}, \mathcal{Q}_{\mathsf{aux}})$ where $\mathsf{st}_{\mathcal{S}}$ is initially set to be the empty string, and $\mathcal{Q}_{\mathsf{sim}}, \mathcal{Q}_{\mathsf{RO}}$ and $\mathcal{Q}_{\mathsf{aux}}$ are initially set to be the empty sets.*

- *$\mathcal{S}_1(\mathbb{x}, \mathsf{aux})$ is an oracle that returns the first output of $\mathcal{S}(1, \mathsf{st}_{\mathcal{S}}, \mathbb{x}, \mathsf{aux})$.*[4]
- *$\mathcal{S}'_1(\mathbb{x}, \mathbb{w})$ is an oracle that first checks $(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$ where pp is part of srs and then runs (and returns the output of) $\mathcal{S}_1(\mathbb{x})$.*
- *$\mathcal{S}_1^F(\mathbb{x}, \mathbb{w})$ is an oracle parameterized by a function F; first, it checks if $(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$, and then runs (and returns the output of) $\mathcal{S}_1(\mathbb{x}, F(\mathbb{x}, \mathbb{w}))$. As explained below, this is useful to model leaky-zero-knowledge.*
- *$\mathcal{S}_2(s, \mathsf{aux})$ is an oracle that first checks if the query s is already present in $\mathcal{Q}_{\mathsf{RO}}$ and in case answers accordingly, otherwise it returns the first output a of $\mathcal{S}(2, \mathsf{st}_{\mathcal{S}}, s)$. Additionally, the oracle updates the set $\mathcal{Q}_{\mathsf{RO}}$ by adding the tuple (s, aux, a) to the set.*

Almost all the oracles in our definitions can take auxiliary information as additional input. We use this auxiliary information in a rather liberal form. For example, in the definition above, the auxiliary information for $\mathcal{S}_1$ refers to the (optional) leakage required by the simulator to work in some cases (see more in Definition 3.4.3), while the auxiliary information for $\mathcal{S}_2$ can contain, for example, the algebraic representations of the group elements in s (when we restrict to algebraic adversaries) or other information the security experiments might need.

[4] More often, simulators need only the first three inputs, see Definition 3.4.2; abusing notation, we assume that such simulators simply ignore the auxiliary input aux.

Definition 3.4.2 (Zero-Knowledge)**.** *A NIZK* NIZK *is (perfect) zero-knowledge if there exists a PPT simulator* $\mathcal{S}$ *such that for all adversaries* $\mathcal{A}$*:*

$$\Pr\left[\begin{array}{l}\mathsf{pp}_{\mathbb{G}} \leftarrow \mathsf{GroupGen}(1^\lambda)\\ \mathsf{srs} \leftarrow \mathsf{KGen}(\mathsf{pp}_{\mathbb{G}})\\ \mathcal{A}^{\mathsf{Prove}(\mathsf{ek},\cdot,\cdot)}(\mathsf{srs}) = 1\end{array}\right] \approx \Pr\left[\begin{array}{l}\mathsf{pp}_{\mathbb{G}} \leftarrow \mathsf{GroupGen}(1^\lambda)\\ (\mathsf{srs},\mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}(0,\mathsf{pp}_{\mathbb{G}})\\ \mathcal{A}^{\mathcal{S}_1'(\cdot,\cdot)}(\mathsf{srs}) = 1\end{array}\right]$$

Zero-knowledge is a security property that is only guaranteed for valid statements in the language, hence the above definition uses $\mathcal{S}_1'$ as a proof simulation oracle.

We also introduce a weaker notion of zero-knowledge. A NIZK is F-leaky zero-knowledge if its proofs may leak some information, namely a proof leaks $F(\mathbb{x},\mathbb{w})$, where $(\mathbb{x},\mathbb{w}) \in \mathcal{R}$. We formalize this by giving the zero-knowledge simulator the value $F(\mathbb{x},\mathbb{w})$, which should be interpreted as a hint for the simulation of proofs. This notion could be seen as an extension of the bounded leaky zero-knowledge property defined in [CFF+21] and tailored for CP-SNARKs. Our notion is a special case of the leakage-resilient zero-knowledge framework of Garg, Jain and Sahai [GJS11] where the leakage of the simulator is known ahead of time.

Definition 3.4.3 (Leaky Zero-Knowledge)**.** *A NIZK* NIZK *is* F*-leaky zero-knowledge if there exists a PPT simulator* $\mathcal{S}$ *such that for all adversaries* $\mathcal{A}$*:*

$$\Pr\left[\begin{array}{l}\mathsf{pp}_{\mathbb{G}} \leftarrow \mathsf{GroupGen}(1^\lambda)\\ \mathsf{srs} \leftarrow \mathsf{KGen}(\mathsf{pp}_{\mathbb{G}})\\ \mathcal{A}^{\mathsf{Prove}(\mathsf{ek},\cdot,\cdot)}(\mathsf{srs}) = 1\end{array}\right] \approx \Pr\left[\begin{array}{l}\mathsf{pp}_{\mathbb{G}} \leftarrow \mathsf{GroupGen}(1^\lambda)\\ (\mathsf{srs},\mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}(0,\mathsf{pp}_{\mathbb{G}})\\ \mathcal{A}^{\mathcal{S}_1^F(\cdot,\cdot)}(\mathsf{srs}) = 1\end{array}\right]$$

3.4.1 Policy-Based Simulation Extractability

An extraction policy defines the constraints under which the extractor must extract the witness. For example, we could consider the policy that checks that the forged instance and proof were not queried/output by the zero-knowledge simulator (thus modeling the classical simulation extractability notion), or we could consider a policy that only checks that the forged instance was not queried to the zero-knowledge simulator, thus obtaining a weaker flavor of classical simulation extractability. Clearly, the more permissive the policy the stronger the security provided.

In our work, we also consider policies that constrain the behavior of the zero-knowledge simulator. For example, we could consider the policy that checks that the queried instances belong to the relation, thus obtaining a notion similar to true-simulation extractability (see Dodis et al. [DHLW10]). Looking ahead, contrary to the true-simulation extractability notion in [DHLW10], our policy-based version of the true-simulation extractability rather than disallowing certain queries, punishes the adversary at extraction time. It is not hard to see that the two definitional flavors, namely disallowing illegal queries versus punishing an adversary that made an illegal query are equivalent in the context of simulation extractability, because the adversary's goal is computational[5].

Extraction policies. We define an extraction policy as a tuple $\Phi = (\Phi_0, \Phi_1)$ of PPT algorithms. This is used to define Φ-simulation extractability as follows. The security experiment

[5]Observe that for decisional tasks disallowing and punishing flavors can result in different security notions, see Bellare, Hofheinz and Kiltz [BHK15].

starts by running the extraction policy algorithm Φ_0, which generates public information pp_Φ. The public information may contain, for example, random values that define the constraints later checked by Φ_1. Therefore, we feed pp_Φ to the adversary. In the case of commit-and-prove proof systems, the public information may contain commitments for which the adversary does not know openings (but on which it can still query simulated proofs). After receiving a forgery from the adversary, the security experiment runs the extraction policy Φ_1. The policy Φ_1 is a predicate that takes as input:

- The public parameter pp_Φ;
- The forged instance and proof $(\mathbb{x}, \pi)$;
- The view of the experiment, denoted view. Such a view contains the public parameters, the set of simulated instances and proofs $\mathcal{Q}_{\mathsf{sim}}$, and the set $\mathcal{Q}_{\mathsf{RO}}$ of queries and answers to the random oracle[6];
- Auxiliary information aux_Φ which might come along with the forged instance. We use aux_Φ to provide the adversary an interface with the policy[7].

Definition 3.4.4 (Φ-Simulation extractability)**.** *Let* Π *be a NIZK for a relation* $\mathcal{R}$ *whose wrappers are* $\mathcal{S}_1, \mathcal{S}_2$*, as defined in Definition 3.4.1 Consider the experiment in Fig. 3.1.* Π *is* Φ*-simulation-extractable (or simply* Φ*-SE) if for every PPT adversary* $\mathcal{A}$ *there exists an efficient extractor* $\mathcal{E}$ *such that the following advantage is negligible in* λ*:*

$$\mathbf{Adv}^{\Phi\text{-}se}_{\Pi,\mathcal{A},\mathcal{S},\mathcal{E}}(\lambda) \coloneqq \Pr\left[\mathbf{Exp}^{\Phi\text{-}se}_{\Pi,\mathcal{A},\mathcal{S},\mathcal{E}}(\lambda) = 1\right]$$

Below, we give a definition that explicitly considers the subclass of PPT algebraic adversaries. To fit algebraic adversaries into our definitional framework we let the algebraic adversaries return the representation vectors (1) for any query to the simulator $\mathcal{S}$ into the auxiliary information aux and (2) for the forgery into the auxiliary information $\mathsf{aux}_{\mathcal{E}}$.

Definition 3.4.5 (Φ-Simulation extractability in the AGM)**.** *Let* Π *be a NIZK for a relation* $\mathcal{R}$ *with a simulator* $\mathcal{S}$*.* Π *is* Φ*-simulation-extractable (or simply* Φ*-SE) if there exists an efficient extractor* $\mathcal{E}$ *such that for every PPT algebraic adversary* $\mathcal{A}$*, the advantage* $\mathbf{Adv}^{\Phi\text{-}se}_{\Pi,\mathcal{A},\mathcal{S},\mathcal{E}}(\lambda)$ *(cf. Definition 3.4.4) is negligible in* λ*.*

3.5 Simulation extractability of KZG in the AGM

3.5.1 CP-SNARK for polynomial evaluation in the AGM

We consider a CP-SNARK $\mathsf{CP}_{\mathtt{evl}}$ for the relation $\mathcal{R}_{\mathtt{evl}}((x, y), f) \coloneqq f(x) = y$, where f is committed as $[f(s)+\alpha r(s)]_1$. The scheme constructed in this section requires one $\mathbb{G}_1$ element

[6]Even if the given NIZK is not in the random oracle (namely neither the prover nor the verifier algorithms make random oracle queries) it still makes sense to assume the existence of the set $\mathcal{Q}_{\mathsf{RO}}$. This is useful to model security for NIZK protocols that eventually are used as sub-protocols in ROM-based protocols (as Universal zkSNARKs based on Polynomial Commitments, see Section 3.6)

[7]For example, looking ahead, in the policy in Section 3.6.3 the adversary that forges a "weak" proof of opening for a commitment, additionally provides a certificate (different from the proof itself) that the commitment is indeed extractable. In this case, we require the extractor only to work for those commitments that come along with valid certificates.

$\mathbf{Exp}^{\Phi\text{-}se}_{\mathcal{A},\mathcal{S},\mathcal{E}}(\lambda)$

$\mathsf{pp}_{\mathbb{G}} \leftarrow\!\$\ \mathsf{GroupGen}(1^\lambda)$
$(\mathsf{srs}, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}(0, \mathsf{pp}_{\mathbb{G}})$
$\mathsf{pp}_\Phi \leftarrow\!\$\ \Phi_0(\mathsf{pp}_{\mathbb{G}})$
$(\mathbb{x}, \pi, \mathsf{aux}_{\mathcal{E}}, \mathsf{aux}_\Phi) \leftarrow \mathcal{A}^{\mathcal{S}_1,\mathcal{S}_2}(\mathsf{srs}, \mathsf{pp}_\Phi)$
$\mathbb{w} \leftarrow \mathcal{E}(\mathsf{srs}, \mathbb{x}, \pi, \mathsf{aux}_{\mathcal{E}})$
$\mathsf{view} \leftarrow (\mathsf{srs}, \mathsf{pp}_\Phi, \mathcal{Q}_{\mathsf{sim}}, \mathcal{Q}_{\mathsf{RO}}, \mathcal{Q}_{\mathsf{aux}})$
if $\Phi_1((\mathbb{x}, \pi), \mathsf{view}, \mathsf{aux}_\Phi) \wedge \mathsf{Verify}^{\mathcal{S}_2}(\mathsf{srs}, \mathbb{x}, \pi)$
$\quad \wedge (\mathsf{pp}, \mathbb{x}, \mathbb{w}) \notin \mathcal{R}$ **then return** 1
else return 0

$\mathcal{S}_1(\mathbb{x}, \mathsf{aux})$:
$\quad \pi, \mathsf{st}_{\mathcal{S}} \leftarrow \mathcal{S}(1, \mathsf{st}_{\mathcal{S}}, \mathbb{x}, \mathsf{aux})$
$\quad \mathcal{Q}_{\mathsf{sim}} \leftarrow \mathcal{Q}_{\mathsf{sim}} \cup \{(\mathbb{x}, \mathsf{aux}, \pi)\}$
$\quad$ **return** π

$\mathcal{S}_2(s, \mathsf{aux})$:
$\quad$ **if** $\nexists \mathsf{aux}, a : (s, \mathsf{aux}, a) \in \mathcal{Q}_{\mathsf{RO}}$:
$\quad\quad a, \mathsf{st}_{\mathcal{S}} \leftarrow \mathcal{S}(2, \mathsf{st}_{\mathcal{S}}, s, \mathsf{aux})$
$\quad\quad \mathcal{Q}_{\mathsf{RO}} \leftarrow \mathcal{Q}_{\mathsf{RO}} \cup \{(s, \mathsf{aux}, a)\}$
$\quad$ **return** a

Figure 3.1: The Φ-simulation extractability experiments in ROM. The extraction policy Φ takes as input the public view of the adversary view (namely, all the inputs received and all the queries and answers to its oracles). The set $\mathcal{Q}_{\mathsf{sim}}$ is the set of queries and answers to the simulation oracle. The set $\mathcal{Q}_{\mathsf{RO}}$ is the set of queries and answers to the random oracle. The set $\mathcal{Q}_{\mathsf{aux}}$ is the set of all the auxiliary information sent by the adversary (depending on the policy, this set might be empty or not). The wrappers $\mathcal{S}_1$ and $\mathcal{S}_2$ deal respectively with the simulation queries and the random oracle queries of $\mathcal{A}$ in the experiment.

to commit to $f(X)$, one $\mathbb{G}_1$ and one $\mathbb{F}_q$ element for the evaluation proof, and checking this proof of evaluation requires two pairings. This is taken from [CFF+21] but adapted to AGM only.

$\mathsf{KGen}_{\mathsf{evl}}$: parse ck as $(([s^j]_1)_{j\in[0,d]}, ([\alpha s^j]_1)_{j\in[0,d]}, [1, s]_2)$ and define $\mathsf{ek} := \mathsf{ck}$ and $\mathsf{vk} := [1, s]_2$, and return $\mathsf{srs} := (\mathsf{ek}, \mathsf{vk})$.

$\mathsf{Prove}_{\mathsf{evl}}(\mathsf{ek}, \mathbb{x} = (\mathsf{c}, x, y), \mathbb{w} = (f, r))$: output $\pi := ([\pi(s)+\alpha\pi'(s)]_1, y')$, where $\pi(X)$ is the polynomial such that $\pi(X)(X - x) \equiv f(X) - y$, and $\pi'(X)$ is such that $\pi'(X)(X - x) \equiv r(X) - r(x)$, and $y' := r(x)$.

$\mathsf{Verify}_{\mathsf{evl}}(\mathsf{vk}, \mathbb{x} = (\mathsf{c}, x, y), (\pi, y'))$: output 1 if and only if:

$$e(\mathsf{c} - [y]_1 - [\alpha y']_1, [1]_2) = e(\pi, [s - x]_2).$$

The above CP-SNARK is knowledge extractable in the AGM [CHM+20]. The original work of [KZG10] proves a weaker notion of security, called evaluation binding, which states that an adversary cannot find two distinct instances (with relative valid proofs) of the form $\mathbb{x} = (\mathsf{c}, x, y)$ and $\mathbb{x}' = (\mathsf{c}, x, y')$. The CP-SNARK supports a bounded (by $\mathsf{deg}(r)$) number of evaluation proofs for a given commitment. One may argue that giving more than $\mathsf{deg}(f)$ evaluations of a polynomial f on distinct points should reveal the polynomial and, thus, the zero-knowledge property would not be needed. However, there are applications in which we could give more than $\mathsf{deg}(f)$ evaluation proofs concerning f without necessarily revealing the evaluation values: e.g., this is achieved when we only show the evaluations of linear combinations of multiple committed polynomials to known constants. Since the technique of [KZG10] would leak information on the

random masking polynomials and would therefore be usable only a limited number of times, one may use the "full-fledged" CP-SNARK of Lunar [CFF+21] for proving an unbounded number of evaluations of committed polynomials in zero-knowledge, at the cost of two additional pairings at verification time.

The scheme that we describe above is zero-knowledge for non-hiding commitments and leaky zero-knowledge (Definition 3.4.3) for hiding commitments. For the latter, given a commitment $\mathsf{c} = [f(s) + \alpha r(s)]_1$, a proof for $\mathbb{x} = (\mathsf{c}, x, y)$ leaks $y' = r(x)$. Thus, we prove that $\mathsf{CP}_{\mathtt{evl}}$ achieves F-leaky zero-knowledge where $F(\mathbb{x} = (\mathsf{c}, x, y), \mathbb{w} = (f, r)) := r(x)$.

We define the simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1)$, where $\mathcal{S}_0$ outputs the trapdoor information s, α together with the srs, and $\mathcal{S}_1$ simulates proofs for $\mathbb{x} = (\mathsf{c}, x, y)$ and leakage y' outputting $\pi = ((\mathsf{c} - [y]_1 - [\alpha y']_1)(s - x)^{-1}, y')$.

Remark 3.5.1. *Consider an attacker that receives a single simulated commitment* c *and queries the ZK simulator twice on the same evaluation point* x_j *with two different evaluation values* y_1 *and* y_2*. These two queries form a linear system without solutions because we have only one variable but two linearly independent equations. These simulation queries lead to the two malleability attacks on KZG that we describe below. For sake of simplicity, we show the attacks in the non-hiding setting.*

(Arbitrary Evaluation Point) *Let* w_1, w_2 *and* c *be such that* $\pi_i = [w_i]_1$ *for* $i \in [2]$*, and* $\mathsf{c}^* = [c]_1$*. We observe that:*

$$(w_1 - w_2)(s - x_j) = y_2 - y_1. \tag{3.1}$$

For an arbitrary point $x^* \neq x_j$*, the adversary sets* $\mathsf{c}^* \leftarrow (\pi_2 - \pi_1)$*,* $\pi^* \leftarrow \frac{\pi_1 - \pi_2}{x^* - x_j}$ *and* $y^* \leftarrow \frac{y_1 - y_2}{x^* - x_j}$*. The tuple* $(\mathsf{c}^*, x^*, y^*, \pi^*)$ *is a valid forgery. Let* w^*, c^* *be such that* $[w^*]_1 = \pi^*$ *and* $[c^*]_1 = \mathsf{c}^*$*. Then:*

$$\begin{aligned} w^*(s - x^*) &= \frac{w_1 - w_2}{x^* - x_j}(s - x^*) = \frac{w_2 - w_1}{x^* - x_j}(x^* - x_j) + \frac{w_1 - w_2}{x^* - x_j}(s - x_j) \\ &= w_2 - w_1 - \frac{y_1 - y_2}{x^* - x_j} = c^* - y^*. \end{aligned}$$

The second equation comes directly from the definition of the elements involved, and the third equation follows because of Eq. (3.1)*.*

(Same Evaluation Point) *For* $x^* = x_j$*, instead, by the homomorphic property of KZG we have* $((\alpha + \beta)\mathsf{c}^*, x_j, \alpha y_1 + \beta y_2)$ *is a valid forgery; we can forge a proof for* c^* *by setting, for example,* $\alpha = 2$ *and* $\beta = -1$*.*

The attacks described in Remark 3.5.1 can be mounted because of the relation between two simulation queries obtained and can be generalized under the notion of attacks that violate the "Algebraic Consistency", as we explain hereafter. Since we focus on CP-SNARKs in which, given a proof π for a statement $\mathbb{x}$, and a view view, it is possible to derive a linear system of polynomial equations $\{p_i(x_j) = y_{i,j}\}_{i,j}$, we introduce the following definition to make the policies easier to describe.

Definition 3.5.1 (Algebraic Consistency)**.** *A view* view *(that might contain a set of simulated instances and proofs for* CP*) satisfies the algebraic consistency for a CP-SNARK* CP *if it is possible to derive (in a way that depends on* CP*) a linear system of polynomial equations that admits a solution.*

The extraction policy for $\mathsf{CP}_{\mathtt{evl}}$. We define $\mathbf{\Phi}^{\mathtt{s\text{-}adpt}}_{\mathtt{evl}} = \{\Phi_{\mathcal{D}}\}_{\mathcal{D}}$ as the family (indexed by a sampler $\mathcal{D}$) of *semi-adaptive* extraction policies for the KZG-based $\mathsf{CP}_{\mathtt{evl}}$ CP-SNARK. Indeed, as we show below, the evaluation points x_j for the instances for which the adversary can see simulated proofs are selectively chosen independently of the commitment key, while the evaluation values y can be adaptively chosen by the adversary. Each policy $\Phi_{\mathcal{D}}$ is a tuple of the form $(\Phi_0^{\mathcal{D}}, \Phi_1)$, as defined in Section 3.4.1, where $\Phi_0^{\mathcal{D}}$ outputs the parameters pp_Φ while Φ_1 outputs a verdict bit. In particular, $\Phi_0^{\mathcal{D}}$ on input group parameters $\mathsf{pp}_{\mathbb{G}}$ outputs $\mathsf{pp}_\Phi := (\mathsf{coms}, \mathcal{Q}_x)$, where coms is a vector of commitments sampled from $\mathcal{D}$, and $\mathcal{Q}_x$ is a set of evaluation points.

For sake of clarity, we define the policy Φ_1 as the logical conjunction of a "simulator" policy Φ_{sim} and an "extractor" policy Φ_{ext}, i.e. $\Phi_1 = \Phi_{\mathsf{sim}} \wedge \Phi_{\mathsf{ext}}$. The first policy defines rules under which we can classify a simulation query *legal*, while the second one defines rules under which the extractor must be able to extract a meaningful witness. We highlight the parts needed only for the hiding setting.

Definition 3.5.2. *Let Φ_{sim} be the policy that returns 1 if and only if:*

1. **Points check:** *let $(\mathbb{x}_i, \mathsf{aux}_i, \pi_i)_i$ be all the entries of $\mathcal{Q}_{\mathsf{sim}}$. Recall that an instance $\mathbb{x}$ can be parsed as (c, x, y). Check that $\forall i : \mathbb{x}_i.x \in \mathcal{Q}_x$.*

2. **Commitment Check:** *For all $i \in [Q_{\mathsf{sim}}]$, parse aux_i as the leakage value y'_i and the representation vectors for $\mathbb{x}_i.\mathsf{c}$ and π_i such that $\vec{r}_i = \vec{f}_i \| \vec{v}_i \| \vec{c}_i$ is the algebraic representation of the commitment $\mathbb{x}_i.\mathsf{c}$. For any i check that $\langle \vec{f}_i \| \vec{v}_i, \mathsf{ek} \rangle + \langle \vec{c}_i, \mathsf{coms} \rangle = \mathbb{x}_i.\mathsf{c}$.*

3. **Algebraic Consistency:** *The simulation queries satisfy the algebraic consistency for $\mathsf{CP}_{\mathtt{evl}}$. Let $\mathcal{I}_j := \{i : \mathbb{x}_i.x = x_j\}$ and let $\vec{R}_j := (\vec{c}_i)_{i \in \mathcal{I}_j}$. Check that $\forall j$: (i) the system of linear equations $\vec{R}_j \cdot \vec{z} = \vec{y}_j$ has at least a solution, where $\vec{z}$ are the variables and $\vec{y}_j = (\mathbb{x}_i.y - \langle \vec{f}_i, (1, x_j, \ldots, x_j^d) \rangle)_{i \in \mathcal{I}_j}$, and (ii) the system of linear equations $\vec{R}_j \vec{z}' = \vec{y}'_j$ has at least a solution, where $\vec{z}'$ are the variables and $\vec{y}'_j = (y'_i - \langle \vec{v}_i, (1, x_j, \ldots, x_j^d) \rangle)_{i \in \mathcal{I}_j}$.*

In more intuitive terms, for every simulation query (c, x, y) made by the adversary: (1) ensures that x is in the set $\mathcal{Q}_x$ chosen at the beginning of the experiment (this is the semi-adaptive restriction); (2) ensures that c is computed as a linear combination of the simulated commitments and the $\mathbb{G}_1$ elements of the SRS, but *not of simulated proofs*; (3) ensures that overall the queried statements are plausibly true (e.g., the adversary does not ask to open the same (c, x) at two different $y \neq y'$).

For the sake of concreteness, we explicitly define the algebraic consistency check in the previous definition. Notice the matrix $\vec{R}_j$ defines linear constraints that must hold for each of the polynomials for which the adversary asks evaluations. In the case of hiding commitments, some of the constraints are obtained by parsing adequately the inputs of the adversary to the simulation oracle.

Next, we define the policy Φ_{ext} as the logical disjunction of two policies, $\Phi^{\mathrm{rnd}}_{\mathsf{ext}}$ and $\Phi^{\mathrm{der}}_{\mathsf{ext}}$. To this end, we first define some notation: let $g_c \colon \mathbb{G}_1 \times \{0,1\}^* \to \{0,1\}$ be a function that on inputs a group element c and a string s, that can be parsed as a list of group elements c_i followed by a second string $\tilde{s}$, outputs 1 if and only if $\exists i : \mathsf{c} = \mathsf{c}_i$.

Definition 3.5.3. *Let $\Phi_{\mathsf{ext}}, \Phi^{rnd}_{\mathsf{ext}}$ and $\Phi^{der}_{\mathsf{ext}}$ be predicates that, parsing the forgery instance $\mathbb{x}^* = (\mathsf{c}^*, x^*, y^*)$, are defined as follows:*

- $\Phi_{\mathsf{ext}}^{rnd}$ *returns 1 if and only if there exist a query* (s, aux, a) *to the random oracle and* aux *contains a non-constant polynomial* $h(X)$ *such that the following conditions are satisfied:*
  1. **Hashing check:** $(s, \mathsf{aux}, a) \in \mathcal{Q}_{\mathsf{RO}}$*, note that* $\mathcal{Q}_{\mathsf{RO}}$ *is contained in* view*,*
  2. **Decoding check:** $g_c(c^*, s) = 1$*.*
  3. **Polynomial check:** $g_h(h, \mathsf{aux}) = 1$*, where* $g_h \colon \mathbb{F}[X] \times \{0,1\}^* \to \{0,1\}$ *is a function that on input a polynomial* $h(X)$ *and a string* aux *outputs 1 if and only if* $h(X)$ *is encoded in* aux*.*
  4. **Computation check:** $h(a) = x^*$*.*
- $\Phi_{\mathsf{ext}}^{der}$ *returns 1 if and only if* $\exists(\mathbb{x}, \cdot, \pi) \in \mathcal{Q}_{\mathsf{sim}}$ *s.t.* $\mathbb{x} \coloneqq (\mathsf{c}^*, x^*, y')$ *and* $(y', \pi) \neq (y^*, \pi^*)$*.*
- Φ_{ext} *returns logical disjunction of* $\Phi_{\mathsf{ext}}^{rnd}$ *and* $\Phi_{\mathsf{ext}}^{der}$*.*

More intuitively, $\Phi_{\mathsf{ext}}^{\mathrm{rnd}}$ checks that the point x^* is obtained from the random oracle *after* querying it on the commitment c^*, whereas $\Phi_{\mathsf{ext}}^{\mathrm{der}}$ checks if $\mathbb{x}^*$ is a strong forgery, namely it is a new evaluation proof for a statement (c^*, x^*) already queried to the simulation oracle.

In the following theorem, we focus on the non-hiding version of KZG, we show in Section 3.5.2 how to handle the hiding setting.

Theorem 3.5.1. *For any witness sampleable distribution* $\mathcal{D}$ *that is* $\mathcal{D}$*-Aff-MDH-secure (see Definition 3.3.1), any bilinear-group generator* $\mathsf{GroupGen}$ *that samples the generator of the group* $\mathbb{G}_1$ *uniformly at random,* $\forall \Phi_{\mathcal{D}} \in \mathbf{\Phi}_{\mathtt{evl}}^{\mathtt{s\text{-}adpt}}$*, the non-hiding KZG is* $\Phi_{\mathcal{D}}$*-simulation-extractable in the AGM. In particular, there exists* $\mathcal{E}$ *such that for any algebraic adversary* $\mathcal{A}$*:*

$$\mathbf{Adv}^{\Phi_{\mathcal{D}}\text{-}\mathsf{se}}_{\mathsf{CP}_{\mathtt{evl}},\mathcal{A},\mathcal{S},\mathcal{E}}(\lambda) \leq O(\epsilon_{(Q_x+d+1)\text{-}\mathsf{DL}}(\lambda)) + O(\epsilon_{\mathit{Aff\text{-}MDH}}(\lambda)) + \mathsf{poly}(\lambda)\epsilon_h$$

where $Q_x \coloneqq |\mathcal{Q}_x|$*,* d *is the maximum degree supported by* $\mathsf{CP}_{\mathtt{evl}}$*,* $\epsilon_{(Q_x+d+1)\text{-}\mathsf{DL}}(\lambda)$ *is the maximum advantage for any algebraic PT adversary against the* $(Q_x + d + 1)$*-strong Discrete-Log Assumption,* $\epsilon_{\mathit{Aff\text{-}MDH}}(\lambda)$ *is the maximum advantage for any algebraic PT adversary against the* $\mathcal{D}$*-Aff-MDH Assumption,* h *is the polynomial that satisfies the Polynomial check of* $\Phi_{\mathcal{D}}$*, and* $\epsilon_h = \frac{\mathtt{deg}(h)}{q}$*.*

Before giving the proof of Theorem 3.5.1, we recall that when $\mathcal{D}$ is the distribution $\mathcal{U}_\ell$ that outputs ℓ uniformly random group elements of $\mathbb{G}_1$ we could reduce the $\mathcal{D}_\ell$-Aff-MDH Assumption to the Discrete Log (see Lemma 3.3.1). Hence, we can state the following corollary, whose proof follows from the fact that $\epsilon_{\mathsf{DL}} \leq \epsilon_{d\text{-}\mathsf{DL}}, \forall d \geq 1$.

Corollary 3.5.1. *For any algebraic adversary* $\mathcal{A}$*, for any* $\ell \in \mathbb{N}$*, and for any distribution* $\mathcal{U}_\ell$ *that outputs* ℓ *uniformly random group elements:*

$$\mathbf{Adv}^{\Phi_{\mathcal{U}_\ell}\text{-}\mathsf{se}}_{\mathsf{CP}_{\mathtt{evl}},\mathcal{A},\mathcal{S},\mathcal{E}}(\lambda) \leq O(\ell\epsilon_{(Q_x+d+1)\text{-}\mathsf{DL}}(\lambda)) + \mathsf{poly}(\lambda)\epsilon_h.$$

Proof intuition of Theorem 3.5.1. We consider an adversary whose forgery satisfies the predicate $\Phi_{\mathsf{ext}}^{\mathrm{rnd}}$. We first show an alternative way to simulate KZG proofs. This step allows one to move from a simulator whose trapdoor is a "secret exponent" s to a simulator whose trapdoor is a "tower" of $\mathbb{G}_1$-elements $[s^i]_1$. The simulated SRS seen by the adversary includes only high-degree polynomials of the form $[p(s)s^i]_1$, while the simulator keeps the low-degree

monomials $[s^i]_1$ for simulation. Here, p is a polynomial that vanishes in all the points to be asked in the simulation queries (this is reminiscent of the reduction technique for Boneh-Boyen signatures [BB04]). Since we program the SRS based on the queries our simulator is only *semi-adaptive*, namely it can simulate proofs for a (exponentially large) subset of all the statements. This first change essentially simplifies the objects involved in our analysis, from rational polynomials (with the formal variable being the trapdoor) to standard polynomials.

Next, we need to show that the adversary cannot mix the simulated commitments and the forgery material. In particular, we need to show that the forged proof is not derived as a linear combination involving simulated commitments. To show this, we use the fact that the degree of the proof must be smaller than the degrees of simulated commitments, otherwise we could break the d-DL assumption in the AGM. This intuitively comes from the fact that the verification equation *lifts* the degree of the polynomial in the forged proof (as it is multiplied by $(X - x^*)$). Similarly, we need to show that the forged instance cannot use a linear combination that involves the simulated commitments. For this, we use the Aff-MDH assumption to handle multiple evaluation proofs on different simulated commitments on the same evaluation point. In particular, we reduce the view of many simulated proofs over many commitments and many evaluation points to a view that only contains $[p(s)s^i]_1$ and (non-rational) polynomials $[p(s)/(s - x_j)]_1$. At this point, the attacker could still perform an attack if it could decide the evaluation point x^* arbitrarily. The attack works as follows: (i) the adversary asks a simulation proof π for $\mathbb{x} = (\mathsf{c}, x, y)$, and (ii) produces the forgery $\mathbb{x}^* = (\mathsf{c} + \alpha\pi, x - \alpha, y), \pi$, for any $\alpha \in \mathbb{Z}_q$. It is easy to check that the forgery satisfies the verification equation. Though, for this attack to work the attacker needs to set the commitment in the forged instance as a function of $x^* = x - \alpha$. The last part of our analysis shows that, indeed, the algebraic representation of the commitment in the forgery cannot depend on x^* and that this attack cannot be mounted when x^* is chosen after the commitment with sufficient randomness.

For the second case, we can reduce a $\Phi^{\mathrm{der}}_{\mathsf{ext}}$ forgery to a $\Phi^{\mathrm{rnd}}_{\mathsf{ext}}$ forgery. In fact, such a forgery together with the simulated proofs set an algebraic inconsistency, a subcase of the condition avoided by Item 3 of Definition 3.5.2, thus enabling an attack. In more detail, given a $\Phi^{\mathrm{der}}_{\mathsf{ext}}$-forgery $(\mathsf{c}, x, y), \pi$ and let $((\mathsf{c}, x, y'), \pi') \in \mathcal{Q}_{\mathsf{sim}}$ we can define a new $\Phi^{\mathrm{rnd}}_{\mathsf{ext}}$-forgery $(\mathsf{c}^*, x^*, y^*), \pi^*$ where $\mathsf{c}^* = (\pi' - \pi)$, $x^* = \mathsf{RO}(\mathsf{c}^*)$ and $\pi^* = \frac{\pi - \pi'}{x^* - x}$ and $y^* = \frac{y - y'}{x^* - x}$. We can prove that the verification equation holds noticing that $(\pi - \pi')(s - x) = [y - y']_1$ and by simple algebraic manipulations.

Proof of Theorem 3.5.1. We stress that $\mathcal{A}$ is algebraic (cf. Definition 2.2.2), therefore for each group element output it additionally attaches a representation $\vec{r}$ of such a group element with respect to all the elements seen during the experiment (included elements in coms). In particular, we assume that for each query $(\mathbb{x}, \mathsf{aux})$ to the oracle $\mathcal{S}_1$ we can parse the value aux as $(\vec{r}, \mathsf{aux}')$ and $\vec{r}$ is a valid representation for $\mathbb{x}.\mathsf{c}$. Similarly, for the queries (s, aux) to $\mathcal{S}_2$, aux includes a valid representation for all the group elements g_i encoded in s, i.e. such that $g_c(\mathsf{g}_i, s) = 1$. Together with its forgery, the algebraic adversary encodes a polynomial $h(X)$ in aux_ϕ, and stores in $\mathsf{aux}_{\mathcal{E}}$ two representation vectors $\vec{r}_{c^*}$ and $\vec{r}_{\pi^*}$ for the two group elements c^* and π^*. We can parse the vectors $\vec{r}_\tau := \vec{f}_\tau \| \vec{c}_\tau \| \vec{o}_\tau$ for $\tau \in \{c^*, \pi^*\}$ where $\vec{f}_\tau$ is the vector of coefficients associated to group elements ek, $\vec{c}_\tau$ is the vector of coefficients associated to group elements $\mathsf{coms} = ([c_i]_1)_{i \in [Q_c]}$, and $\vec{o}_\tau$ is the vector of coefficients associated to the group elements

of the simulated proofs proofs. Namely, we have:

$$\mathsf{c}^* = \langle \vec{f}_{c^*}, \mathsf{ek} \rangle + \langle \vec{c}_{c^*}, \mathsf{coms} \rangle + \langle \vec{o}_{c^*}, \mathsf{proofs} \rangle \tag{3.2}$$

$$\pi^* = \langle \vec{f}_{\pi^*}, \mathsf{ek} \rangle + \langle \vec{c}_{\pi^*}, \mathsf{coms} \rangle + \langle \vec{o}_{\pi^*}, \mathsf{proofs} \rangle \tag{3.3}$$

We can assume w.l.g. that all the simulation queries and the forgery of the adversary $\mathcal{A}$ agree with the policy $\Phi_{\mathcal{D}}$, as otherwise the adversary would automatically lose the experiment. We assume that $\vec{f}_{i,j} = \vec{0}, \forall i, j$, i.e., the adversary asks simulation queries on commitments that are a linear combination of coms only: this is also w.l.g. as we briefly show below. Given a commitment $\mathsf{c}_{i,j} = \mathbb{x}_{i,j}.\mathsf{c}$, whose representation is $\vec{r}_{i,j} = \vec{f}_{i,j} \| \vec{c}_{i,j}$, the adversary could compute a proof $\pi_{i,j}$ for the point x_j and the evaluation value y as follows:

1. let $y' = f_{i,j}(x_j)$, $\mathcal{A}$ computes the commitment $\mathsf{c}' \leftarrow \mathsf{Com}(\mathsf{ck}, f_{i,j}(X))$, and the "honest" proof π' for (c', x_j, y')
2. asks the simulation oracle to provide a proof $\tilde{\pi}$ for the instance $(\mathsf{c} - \mathsf{c}', x_j, y - y')$ with representation $\vec{0} \| \vec{c}_{i,j}$
3. recombines the proof $\pi_{i,j} = \pi' + \tilde{\pi}$

We define our extractor to be the *canonical* extractor that returns the polynomial $f(X) \leftarrow \langle \vec{f}_{c^*}, (1, X, \ldots, X^d) \rangle$.

We start by proving that for any algebraic adversary $\mathcal{A}$ whose forgery satisfies the predicate $\Phi^{\mathrm{der}}_{\mathsf{ext}}$, there exists an algebraic adversary $\mathcal{B}$ whose forgery satisfies the predicate $\Phi^{\mathrm{rnd}}_{\mathsf{ext}}$. Let $\{\Phi'_{\mathcal{D}}\}_{\mathcal{D}}$ be the family of policies defined exactly as $\mathbf{\Phi}^{\mathsf{s\text{-}adpt}}_{\mathsf{evl}}$ with the difference that the extraction policy Φ_{ext} is equal to $\Phi^{\mathrm{rnd}}_{\mathsf{ext}}$ (i.e., there is no logical disjunction with $\Phi^{\mathrm{der}}_{\mathsf{ext}}$).

Lemma 3.5.1. *For any algebraic adversary $\mathcal{A}$ there exists an algebraic adversary $\mathcal{B}$ such that:*

$$\mathbf{Adv}^{\Phi_{\mathcal{D}}\text{-}\mathsf{se}}_{\mathsf{CP}_{\mathsf{evl}},\mathcal{A},\mathcal{S},\mathcal{E}}(\lambda) = \mathbf{Adv}^{\Phi'_{\mathcal{D}}\text{-}\mathsf{se}}_{\mathsf{CP}_{\mathsf{evl}},\mathcal{B},\mathcal{S},\mathcal{E}}(\lambda)$$

Proof. First, we notice that once we fix a commitment c, a point x, and a value y, there is a unique proof π that can satisfy the KZG verification equation. Thus, the predicate $\Phi^{\mathrm{der}}_{\mathsf{ext}}$ can be simplified as requiring that an adversary outputs a valid proof π^* and a value y^* such that $\exists((\mathsf{c}^*, x^*, y'), \cdot, \pi) \in \mathcal{Q}_{\mathsf{sim}}$ and $y^* \neq y'$.

The reduction $\mathcal{B}$ internally runs $\mathcal{A}$ forwarding all the simulation queries, up to the forgery $(\mathbb{x}^*, \pi^*)$, where $\mathbb{x}^* = (\mathsf{c}^*, x^*, y^*)$. If the simulation queries and/or the forgery of the adversary $\mathcal{A}$ do not agree with the policy $\Phi_{\mathcal{D}}$, i.e. $\mathcal{A}$ automatically loses its game, $\mathcal{B}$ aborts. Otherwise, it must be true that the forgery of $\mathcal{A}$ either (i) satisfies the extraction predicate $\Phi^{\mathrm{rnd}}_{\mathsf{ext}}$ or (ii) satisfies the extraction predicate $\Phi^{\mathrm{der}}_{\mathsf{ext}}$. Both cases can be efficiently checked by $\mathcal{B}$. In case (i) $\mathcal{B}$ would simply forward the forgery of $\mathcal{A}$ retaining the same advantage of $\mathcal{A}$. Otherwise, before submitting the forgery, $\mathcal{B}$ retrieves from $\mathcal{Q}_{\mathsf{sim}}$ the statement $\mathbb{x} \coloneqq (\mathsf{c}^*, x^*, y')$, where $y' \neq y^*$, and the corresponding proof π output by $\mathcal{S}_1$. Then $\mathcal{B}$ produces the forgery:

$$\hat{c} \leftarrow \pi^* - \pi, \qquad \hat{x} \leftarrow h(a), \qquad \hat{\pi} \leftarrow \frac{\pi - \pi^*}{\hat{x} - x^*}, \qquad \hat{y} \leftarrow \frac{y' - y^*}{\hat{x} - x^*}$$

which satisfies the verification equation (cf. Remark 3.5.1), and the extraction predicate $\Phi^{\mathrm{rnd}}_{\mathsf{ext}}$ when $(\hat{c}, h, a) \in \mathcal{Q}_{\mathsf{RO}}$. □

Thanks to Lemma 3.5.1 we can assume that the forgery of $\mathcal{A}$ satisfies the extraction predicate $\Phi^{\mathrm{rnd}}_{\mathsf{ext}}$. We let $\mathbf{H}_0$ be the $\mathbf{Exp}^{\Phi_{\mathcal{D}}\text{-}\mathsf{se}}_{\mathcal{A},\mathcal{S},\mathcal{E}}(\lambda)$ experiment, and we denote by ϵ_i the advantage of $\mathcal{A}$ to win $\mathbf{H}_i$, i.e. $\epsilon_i \coloneqq \Pr[\mathbf{H}_i = 1]$.

Hybrid $\mathbf{H}_1$. Recall that $\mathcal{D}$ is witness sampleable, thus according to Definition 2.3.3 there exists a PPT algorithm $\tilde{\mathcal{D}}$ associated with the sampler $\mathcal{D}$. The hybrid experiment $\mathbf{H}_1$ is identical to the previous one, but the group elements in coms are "sampled at exponent", i.e. we use $\tilde{\mathcal{D}}$ to generate the field elements $\vec{\gamma}$, and we let $\mathsf{coms} \leftarrow [\vec{\gamma}]_1$; we also add $\vec{\gamma}$ to $\mathsf{st}_{\mathcal{S}}$. By the witness sampleability of $\mathcal{D}$, $\mathbf{H}_0$ and $\mathbf{H}_1$ are perfectly indistinguishable, thus $\epsilon_1 = \epsilon_0$.

Hybrid $\mathbf{H}_2$. In this hybrid, we change the way we generate the SRS srs and the way in which $\mathcal{S}_1$ simulates the proofs.

Let $\left((\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), [1]_1, [1]_2\right) \leftarrow\!\$\ \mathsf{GroupGen}(1^\lambda)$, sample $s \leftarrow\!\$\ \mathbb{F}$ and compute $\left[s, \ldots, s^{D+d}\right]_1$, $[1, s]_2$, where $D \leftarrow Q_x + 1$. Let $x_r \leftarrow\!\$\ \mathbb{F}$, and let $p(X)$ be the vanishing polynomial in $\mathcal{Q}_x \cup \{x_r\}$, namely:

$$p(X) \coloneqq (X - x_r) \prod_{x \in \mathcal{Q}_x} (X - x).$$

Also, let $p_j(X) \coloneqq p(X)(X - x_j)^{-1}$, for $j \in [Q_x]$. In $\mathbf{H}_2$ we have that:

- $\mathsf{pp}_{\mathbb{G}} \coloneqq ((\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), [p(s)]_1, [1]_2)$,
- $\mathsf{srs} \coloneqq (\mathsf{ek}, \mathsf{vk})$, where $\mathsf{ek} \leftarrow \left[p(s), p(s)s, \ldots, p(s)s^d\right]_1$ and $\mathsf{vk} \leftarrow [1, s]_2$,
- $\mathsf{st}_{\mathcal{S}} \coloneqq \left[1, s, \ldots, s^{D+d}\right]_1, [1, s]_2, \vec{\gamma}$.

Upon a query of the form $(\mathbb{x} = (\mathsf{c}, x_j, y_k), \mathsf{aux} = (\vec{r}_c, \mathsf{aux}'))$ to $\mathcal{S}_1$, the latter outputs the proof $\pi \leftarrow [(\langle \vec{r}_c, \vec{\gamma} \rangle - y_k) \cdot p_j(s)]_1$, and updates $\mathcal{Q}_{\mathsf{sim}}$ accordingly.

We now show that $\mathbf{H}_1 \equiv \mathbf{H}_2$, i.e., the view offered to the adversary $\mathcal{A}$ is identically distributed in the two experiments.

Lemma 3.5.2. $\epsilon_2 = \epsilon_1$.

Proof. Notice that in $\mathbf{H}_2$ we sample from $\mathsf{GroupGen}$ the description of the group, and then we set the generator of $\mathbb{G}_1$ to $[p(s)]_1$ which, thanks to the random root x_r, is distributed uniformly at random even given the value s. It is not hard to verify that the simulated proofs generated by the hybrid $\mathbf{H}_2$ pass the verification equations, in fact, we are assuming that queried commitment c are of the form $\langle \vec{r}_c, \mathsf{coms} \rangle$. Additionally, since the proofs are uniquely determined given the SRS and the statements, the simulated proofs created in $\mathbf{H}_2$ are distributed as the simulated proofs generated by the simulator $\mathcal{S}_1$ in $\mathbf{H}_1$. Thus, the advantage of $\mathcal{A}$ is the same in the two experiments. □

Given an algebraic adversary $\mathcal{A}$ we can define a new adversary, $\mathcal{A}_{\mathsf{c}}$, that we call the *core* adversary. Whenever the adversary $\mathcal{A}$ outputs a group element g it provides a representation vector $\vec{r}_{\mathsf{g}} \coloneqq \vec{f}_{\mathsf{g}} \| \vec{c}_{\mathsf{g}} \| \vec{o}_{\mathsf{g}}$ for g such that:

$$\mathsf{g} = \langle \vec{f}_{\mathsf{g}}, \mathsf{ek} \rangle + \langle \vec{c}_{\mathsf{g}}, \mathsf{coms} \rangle + \langle \vec{o}_{\mathsf{g}}, \mathsf{proofs} \rangle.$$

The adversary $\mathcal{A}_{\mathsf{c}}$ runs internally $\mathcal{A}$ and forwards all the queries and answers from $\mathcal{A}$ to its simulation oracle. However, the way of simulating RO queries must ensure to not alter the

result of the extractor policy, i.e. the "hash-check" for x^*. This is why we cannot simply forward the queries of $\mathcal{A}$ to the random oracle. Therefore, we keep track of the queries made by $\mathcal{A}$ in the list $\mathcal{Q}_{\mathsf{RO},\mathcal{A}}$ and the list of queries made by the core adversary in $\mathcal{Q}_{\mathsf{RO}}$. More in detail, when $\mathcal{A}$ queries the RO with (s, aux), the adversary $\mathcal{A}_{\mathsf{c}}$ makes a "core" RO query $(s_{\mathsf{c}}, \mathsf{aux}_{\mathsf{c}})$ such that:

1. Let s be parsed as $(\mathsf{g}_i)_{i\in[k]}$ (the group elements in s whose representations $\vec{r}_{\mathsf{g}_i} \coloneqq \vec{f}_{\mathsf{g}_i} \| \vec{c}_{\mathsf{g}_i} \| \vec{o}_{\mathsf{g}_i}$ are in aux) and a string $\tilde{s}$. Notice, since the adversary is algebraic we can unambiguously parse s as such.

2. For each i, $\mathcal{A}_{\mathsf{c}}$ computes the group elements $\mathsf{g}'_i = \mathsf{g}_i - \langle \vec{f}_{\mathsf{g}_i}, \mathsf{ek} \rangle$. $\mathcal{A}_{\mathsf{c}}$ encodes into the string s' the group elements $(\mathsf{g}_i, \mathsf{g}'_i)_{i\in[k]}$.

3. $\mathcal{A}_{\mathsf{c}}$ queries the RO with $(s_{\mathsf{c}}, \mathsf{aux}_{\mathsf{c}})$, where $s_{\mathsf{c}} \coloneqq s' \| \tilde{s}$, and $\mathsf{aux}_{\mathsf{c}}$ contains the representations of all the group elements in s' and the same function h encoded in aux. Finally, it forwards the output to $\mathcal{A}$, i.e. it adds (s, aux, a) to $\mathcal{Q}_{\mathsf{RO},\mathcal{A}}$, and adds (s, s_{c}) to (the initially empty) $\mathcal{Q}_s$.

Eventually, $\mathcal{A}$ outputs as forgery a string s and the tuple $(\mathsf{c}', x', y', \pi')$, together with representation vectors $\vec{r}_{c'}$ and $\vec{r}_{\pi'}$. Let $f(X) \coloneqq \langle \vec{f}_{c'}, (1, X, \ldots, X^d) \rangle$, $y \coloneqq f(x')$, and $q(X)$ be such that $q(X)(X - x') = f(X) - y$. Let $\vec{f}_q$ be the vector of the coefficients of $q(X)$, namely $q(X) \coloneqq \langle \vec{f}_q, (1, X, \ldots, X^d) \rangle$. The core adversary $\mathcal{A}_{\mathsf{c}}$ returns for its forgery the string s_{c} such that $(s, s_{\mathsf{c}}) \in \mathcal{Q}_s$, and the tuple $(\mathsf{c}^*, x', y^*, \pi^*)$, where $y^* \leftarrow y' - f(x')$ and:

$$\mathsf{c}^* \leftarrow \mathsf{c}' - \underbrace{[f(s)p(s)]_1}_{\mathsf{Com}(\mathsf{ck}, f(X))}, \qquad \pi^* \leftarrow \pi' - \underbrace{[q(s)p(s)]_1}_{\mathsf{Com}(\mathsf{ck}, q(X))}$$

inserting into aux_Φ the (correct) algebraic representations $(\vec{0} \| \vec{c}_{c'} \| \vec{o}_{c'})$ for c^* and $((\vec{f}_{\pi'} - \vec{f}_q) \| \vec{c}_{\pi'} \| \vec{o}_{\pi'})$ for π^*.

Hybrid $\mathbf{H}_3$. This hybrid is exactly the same of $\mathbf{H}_2$ but instead of running the experiment with the adversary $\mathcal{A}$ we run it with the core adversary $\mathcal{A}_{\mathsf{c}}$.

Lemma 3.5.3. $\epsilon_3 = \epsilon_2$.

Proof. First, by construction, it is easy to verify that $\mathcal{A}_{\mathsf{c}}$ is algebraic. Thus, we need to show that the forgery of $\mathcal{A}$ is valid if and only if the forgery of $\mathcal{A}_{\mathsf{c}}$ is valid. By construction, we have:

$$\mathsf{c}^* \coloneqq \mathsf{c}' - [f(s)p(s)]_1 , \quad \pi^* \coloneqq \pi' - [q(s)p(s)]_1 , \quad y^* \coloneqq y' - f(x^*).$$

By the verification equation of the forgery of $\mathcal{A}_{\mathsf{c}}$ we have:

$$\begin{aligned}
&e(\mathsf{c}^* - [y^*]_1 , [1]_2) - e(\pi^*, [s - x^*]_2) = \\
&e(\mathsf{c}' - [f(s)p(s)]_1 - [y' - f(x')]_1 , [1]_2) - e(\pi' - [q(s)p(s)]_1 , [s - x^*]_2) = \\
&e(\mathsf{c}' - [y']_1 , [1]_2) - e(\pi', [s - x']_2) - [f(s)p(s) - f(x') - q(s)p(s)(s - x^*)]_T = \\
&e(\mathsf{c}' - [y']_1 , [1]_2) - e(\pi', [s - x']_2),
\end{aligned}$$

where the last equation holds since $q(X)(X - x') = (f(X) - f(x'))$ and $x^* = x'$.

Finally, notice that a forgery is valid for $\mathcal{A}$ if it provides a string s that satisfies the "hash check" of Φ_{ext}. We have that there exist s, aux, a, and $h(X)$ such that: (i) $g_c(\mathsf{c}^*, s) = 1$, (ii) $g_h(h, \mathsf{aux}) = 1$, (iii) $(s, \mathsf{aux}, a) \in \mathcal{Q}_{\mathsf{RO},\mathcal{A}}$, and (iv) $x^* = h(a)$ for the forgery of $\mathcal{A}$.

The way $\mathcal{A}_{\mathsf{c}}$ simulates the RO queries ensures that for the query s of $\mathcal{A}$ to the RO, the core adversary sent the "core" RO query s_{c} that encodes both c' and c^*, thus we have that (i) $g_c(\mathsf{c}^*, s_{\mathsf{c}}) = 1$, (ii) $g_h(h, \mathsf{aux}_{\mathsf{c}}) = 1$, (iii) $(s_{\mathsf{c}}, \mathsf{aux}_{\mathsf{c}}, a) \in \mathcal{Q}_{\mathsf{RO}}$, and (iv) $x^* = h(a)$ for the forgery of $\mathcal{A}_{\mathsf{c}}$. □

Notice that if we run the canonical extractor on the outputs of the core adversary $\mathcal{A}_{\mathsf{c}}$, the extractor sets the extracted witness to be the zero polynomial.

Hybrid $\mathbf{H}_4$. The hybrid $\mathbf{H}_4$ additionally checks that $\vec{f}_{\pi^*} \neq \vec{0} \vee \vec{c}_{\pi^*} \neq \vec{0}$, and if the condition holds the adversary $\mathcal{A}_{\mathsf{c}}$ loses the game.

Lemma 3.5.4. $\epsilon_3 \leq \epsilon_4 + \epsilon_{(Q_x+d+1)\text{-DL}}$

Proof. Recall that from the definition of the experiment, upon a query $(\mathbb{x}, \mathsf{aux})$ from $\mathcal{A}_{\mathsf{c}}$ to the simulation oracle of the form $\mathbb{x} = (\mathsf{c}, x_j, y_k)$ and $\mathsf{aux} = \vec{r}$ where $\mathsf{c} = \langle \vec{r}, \mathsf{coms} \rangle$, the adversary receives the proof $[\pi_{\vec{r},j,k}(s)]_1$ where:

$$\pi_{\vec{r},j,k}(X) \coloneqq (\langle \vec{r}, (\gamma_i)_i \rangle - y_k) p_j(X).$$

Consider the following polynomials:

$$\begin{aligned}
c^*(X) &\coloneqq \sum_{i \in [Q_c]} c_{c^*,i} \cdot \gamma_i p(X) + \sum_{\vec{r},j,k} o_{c^*,\vec{r},j,k} \cdot \pi_{\vec{r},j,k}(X) \\
\pi^*(X) &\coloneqq \sum_{i \in [Q_c]} c_{\pi^*,i} \cdot \gamma_i p(X) + \sum_{\vec{r},j,k} o_{\pi^*,\vec{r},j,k} \cdot \pi_{\vec{r},j,k}(X) + \sum_{i \in [d+1]} f_{\pi^*,i} X^{i-1} p(X) \\
v(X) &\coloneqq c^*(X) - y^* p(X) - (X - x^*)\pi^*(X)
\end{aligned}$$

By the guarantees of the AGM, we have $\mathsf{c}^* = [c^*(s)]_1$ and $\pi^* = [\pi^*(s)]_1$, moreover, if the verification equation is satisfied by the forgery of $\mathcal{A}_{\mathsf{c}}$, then $v(s) = 0$.

Next, we show that when the forgery of the adversary is valid the probability of $\vec{f}_{\pi^*} \neq \vec{0}$ or $\vec{c}_{\pi^*} \neq \vec{0}$ is bounded by $\epsilon_{(Q_x+d+1)\text{-DL}}$.

First, notice that if the verification equation for $\mathcal{A}_{\mathsf{c}}$ holds then the polynomial $v(X)$ must be equivalent to the zero polynomial with overwhelming probability. In fact, $v(s) = 0$ when the verification equation holds; if $v(X)$ is not the zero polynomial then, by Lemma 3.3.2, we can reduce $\mathcal{A}_{\mathsf{c}}$ to an adversary to the $(Q_x + d + 1)$-DL assumption. Thus:

$$c^*(X) - y^* p(X) - (X - x^*)\pi^*(X) = v(X) = 0. \tag{3.4}$$

By the guarantees of the AGM, the polynomial $\pi^*(X)$ is a linear combination of elements that depend on $X^{i-1}p(X)$ for $i \in [d+1]$ and $p_j(X)$ for $j \in [Q_x]$. However, when the verification equation holds, the degree of $\pi^*(X)$ must be strictly less than the degree of $p(X)$, because, by Eq. (3.4), $v(X)$ would contain a non-zero coefficient of degree $Q_x + d + 1$ which in particular implies that $v(X) \not\equiv 0$. Then it must be the case that the forged proof $\pi^*(s)$ is a linear combination of the simulated proofs only, thus both $\vec{f}_{\pi^*}$ and $\vec{c}_{\pi^*}$ are null. □

The representation of c^* and π^* computed by the adversary (possibly) depends on the elements $\pi_{\vec{r},j,k}$ (i.e. the proof for the linear combination $\vec{r}$ of the elements of coms with evaluation point x_j and evaluation value y_k) of proofs. However, it is much more convenient to give a representation that depends on the polynomials $p_j(X)$. This motivates the definition of our next hybrid.

Hybrid $\mathbf{H}_5$. The hybrid $\mathbf{H}_5$ finds coefficients $\vec{o}''_\tau$, for $\tau \in \{c^*, \pi^*\}$ such that:

$$\langle \vec{o}_\tau, \mathsf{proofs} \rangle = \langle \vec{o}''_\tau, ([p_j(s)]_1)_j \rangle. \tag{3.5}$$

Moreover, if $\vec{o}_{c^*} \neq \vec{0}$ but $\vec{o}''_{c^*} = \vec{0}$ the adversary loses the game.

Lemma 3.5.5. $\epsilon_4 \leq \epsilon_5 + \epsilon_{\textit{Aff-MDH}}$

Proof. We begin by showing that the hybrid can compute such alternative representations efficiently. We proceed in steps.

Let us parse the simulated proofs $\mathsf{proofs} \coloneqq (\pi_{j,\ell})_{j,\ell}$ such that $\pi_{j,\ell}$ is the ℓ-th simulated proof obtained by $\mathcal{S}_1$ on a query involving the j-th point x_j, i.e., $((x_j, \hat{\mathsf{c}}_{j,\ell}, y_{j,\ell}), \mathsf{aux}_{j,\ell})$. Also, let $\vec{c}_{j,\ell}$ be the algebraic representation for the group element $\hat{\mathsf{c}}_{j,\ell}$ in $\mathsf{aux}_{j,\ell}$. For every $j \in [Q_x]$, we define $\vec{R}_j$ as the $Q_c \times Q_c$ matrix whose ℓ-th column is $\vec{c}_{j,\ell}$.

By construction of $\mathcal{S}_1$ in this hybrid we have that for every $j \in [Q_x]$ it holds

$$\pi_{j,\ell} \coloneqq \left[(\vec{c}_{j,\ell}^{\top} \cdot \vec{\gamma} - y_{j,\ell})\, p_j(s) \right]_1$$

and thus $\tilde{\pi}_j \coloneqq \left[(\vec{R}_j^{\top} \vec{\gamma} - \vec{y}_j) p_j(s) \right]_1$ with $\vec{y}_j \coloneqq (y_{j,\ell})_\ell$.

Without loss of generality, we assume that for each x_j the adversary makes the maximum number of simulation queries (i.e., $\ell \in [Q_c]$); therefore $\vec{R}_j$ is a full rank matrix (this follows from the fact that the simulation queries of the adversary satisfy the policy Φ_{sim}, and in particular the algebraic consistency of the policy, see Item 3).

Given any vector $\vec{o}_\tau$ with $\tau \in \{\mathsf{c}^*, \pi^*\}$, its m-th entry $o_{\tau,m}$ corresponds to the m-th simulated proof in proofs. Therefore, similarly to above, we denote by $o_{\tau,j,\ell}$ the entry corresponding to proof $\pi_{j,\ell}$ and we define $\vec{o}_{\tau,j} \coloneqq (o_{\tau,j,\ell})_\ell$.

Then, for every $j \in [Q_x]$ we define

$$\vec{o}'_{\tau,j} \leftarrow \vec{R}_j \cdot \vec{o}_{\tau,j} \tag{3.6}$$

$$\vec{\pi}'_j \leftarrow (\vec{R}_j^{\top})^{-1} \cdot \vec{\pi}_j \tag{3.7}$$

from which we derive that for any τ:

$$\begin{aligned} \sum_j \langle \vec{o}'_{\tau,j}, \vec{\pi}'_j \rangle &= \sum_j \langle \vec{R}_j \cdot \vec{o}_{\tau,j}, (\vec{R}_j^{\top})^{-1} \cdot \vec{\pi}_j \rangle \\ &= \sum_j \vec{o}_{\tau,j}^{\top} \cdot \vec{R}_j^{\top} (\vec{R}_j^{\top})^{-1} \cdot \vec{\pi}_j \\ &= \sum_j \langle \vec{o}_{\tau,j}, \vec{\pi}_j \rangle \end{aligned}$$

which is equal to $\langle \vec{o}_\tau, \mathsf{proofs} \rangle$, up to a permutation of the indices j.

For all $j \in [Q_x]$ let $\vec{z}_j \coloneqq (\vec{R}_j^\top)^{-1} \cdot \vec{y}_j$, and note that

$$\vec{\pi}'_j = [(\vec{\gamma} - \vec{z}_j)p_j(s)]_1 ,$$

namely $\pi'_{j,i}$ is a valid proof for the instance $(\mathsf{c}_i, x_j, z_{j,i})$ w.r.t. the simulated SRS.

The hybrid $\mathbf{H}_5$ computes $o''_{\tau,j} \leftarrow \langle \vec{o}'_{\tau,j}, (\vec{\gamma} - \vec{z}_j)\rangle$, and $\vec{o}''_\tau \leftarrow (o''_{\tau,j})_{j\in[Q_x]}$. By construction:

$$\sum_{j\in[Q_x]} \langle \vec{o}'_{\tau,j}, \vec{\pi}'_j\rangle = \sum_{j\in[Q_x]} o''_{\tau,j} \cdot [p_j(s)]_1$$

which proves the first part of the lemma, i.e., computing $\vec{o}''_{\tau,j}$ satisfying Eq. (3.5).

In what follows, we prove that if the event that $\mathbf{H}_5$ outputs 0 but $\mathbf{H}_4$ would output 1, namely that all the conditions of $\mathbf{H}_4$ hold but $\vec{o}_{\mathsf{c}^*} \neq \vec{0} \wedge \vec{o}''_{\mathsf{c}^*} = \vec{0}$, then we can break the Aff-MDH assumption.

First, notice that for any j $\vec{o}_{\mathsf{c}^*,j} \neq \vec{0}$ implies that $\vec{o}'_{\mathsf{c}^*,j} \neq \vec{0}$, because the linear transformation applied to compute $\vec{o}'_{\mathsf{c}^*,j}$ is full rank. Second, take an index j^* such that $\vec{o}_{\mathsf{c}^*,j^*} \neq \vec{0}$ and set $\vec{A} \leftarrow \vec{o}'_{\mathsf{c}^*,j^*}$ and $\zeta \leftarrow \langle \vec{z}_{j^*}, \vec{o}'_{\mathsf{c}^*,j^*}\rangle$.

By the above definition of the values o''_{c^*,j^*} and our assumption that the "bad event" of this hybrid is $\vec{o}''_{\mathsf{c}^*} = \vec{0}$, we have that:

$$\langle \vec{A}, [\vec{\gamma}]_1\rangle = [\underbrace{\langle \vec{o}'_{\mathsf{c}^*,j^*}, (\vec{\gamma} - \vec{z}_{j^*})\rangle}_{o''_{\mathsf{c}^*,j^*}=0}]_1 + [\underbrace{\langle \vec{o}'_{\mathsf{c}^*,j^*}, \vec{z}_{j^*}\rangle}_{\zeta}]_1 = [\zeta]_1 .$$

The reduction $\mathcal{B}$ to the $\mathcal{D}$-Aff-MDH Assumption takes as input a distribution $[\vec{\gamma}]_1$ and runs the experiment as in $\mathbf{H}_4$ (it perfectly emulates $\mathbf{H}_4$, and in particular the simulation oracle, because it knows the trapdoor s "at the exponent"). Then $\mathcal{B}$ computes the coefficients $(A_i)_{i\in[Q_c]}$ and the value ζ as described above, which is a valid $\mathcal{D}$-Aff-MDH solution. □

Hybrid $\mathbf{H}_6$. The hybrid $\mathbf{H}_6$ additionally checks that $\vec{r}_{c^*} \neq \vec{0}$, and if the condition holds the adversary $\mathcal{A}_\mathsf{c}$ loses the game.

Lemma 3.5.6. $\epsilon_5 \leq \epsilon_6 + \epsilon_{\mathit{Aff\text{-}MDH}} + 2\epsilon_{(Q_x+1+d)\text{-}\mathsf{DL}} + \mathsf{poly}(\lambda)\frac{\deg(h)}{q}$

Proof. We bound the probability that the adversary loses in $\mathbf{H}_6$ but not in $\mathbf{H}_5$, namely, the probability that $\vec{r}^*_c \neq \vec{0}$ but the conditions of $\mathbf{H}_5$ hold. We show a reduction $\mathcal{B}$ to the Aff-MDH when this event happens.

First, we can assume that the core adversary outputs coefficients $\vec{f}_{c^*} = \vec{f}_{\pi^*} = \vec{c}_{\pi^*} = \vec{0}$, i.e. the adversary only makes use of previous commitments $\mathsf{c}_i \in \mathsf{coms}$ and simulated proofs $\pi_{\vec{r},j,k} \in \mathsf{proofs}$ to represent c^*, and only uses the simulated proofs to represent the proof π^*.

The reduction $\mathcal{B}$ takes as input a distribution $[\vec{\gamma}]_1$ and runs the experiment as in $\mathbf{H}_5$. $\mathcal{B}$ aborts if the forgery $(\mathsf{c}^*, x^*, y^*, \pi^*)$ returned by the adversary is not valid (i.e. either the extraction predicate or the verification equation is not satisfied) or $\vec{r}_{c^*} = \vec{0}$. Otherwise, we have that:

$$e(\mathsf{c}^* - [p(s)y^*]_1 , [1]_2) = e(\pi^*, [s - x^*]_2) \text{ and } \vec{r}_{c^*} \neq \vec{0}$$

where $\vec{r}_{c^*} \neq \vec{0}$ if $\vec{o}_{c^*} \neq \vec{0} \vee \vec{c}_{c^*} \neq \vec{0}$.

We can then rewrite the commitment and the proof of forgery of the core adversary as a function of the coefficients $\vec{o}''_{c^*}$ and $\vec{o}''_{\pi^*}$ (as computed in the $\mathbf{H}_5$):

$$\mathsf{c}^* \coloneqq \sum_{i\in[Q_c]} c_{c^*,i}\left[\gamma_i p(s)\right]_1 + \sum_{j\in[Q_x]} o''_{c^*,j}\left[p_j(s)\right]_1, \qquad \pi^* \coloneqq \sum_{j\in[Q_x]} o''_{w^*,j}\left[p_j(s)\right]_1$$

Since the verification equation is satisfied, and plugging in the AGM representations we have:

$$\sum_{i\in[Q_c]} c_{c^*,i}\gamma_i p(s) + \sum_{j\in[Q_x]} o''_{c^*,j}p_j(s) - p(s)y^* = \sum_{j\in[Q_x]} o''_{\pi^*,j}p_j(s)(s-x^*) \tag{3.8}$$

For all $j \in [Q_x]$, we define $\delta_j \coloneqq x_j - x^*$. We can rewrite the r.h.s. of Eq. (3.8) as:

$$\begin{aligned}\sum_{j\in[Q_x]} o''_{\pi^*,j}p_j(s)(s-x^*) &= \sum_{j\in[Q_x]} o''_{\pi^*,j}p_j(s)((s-x_j)+\delta_j)) \\ &= \sum_{j\in[Q_x]} o''_{\pi^*,j}(p(s)+p_j(s)\delta_j)\end{aligned}$$

In Eq. (3.8), we group all the terms that depend on $p(s)$ on the left side, and we move all the terms that depend on $p_j(s)$ to the right side, thus obtaining:

$$\underbrace{\left(\sum_{i\in[Q_c]} c_{c^*,i}\gamma_i - \sum_{j\in[Q_x]} o''_{w^*,j} - y^*\right)}_{A} p(s) = \sum_{j\in[Q_x]} \underbrace{\left(o''_{w^*,j}\delta_j - o''_{c^*,j}\right)}_{B_j} p_j(s) \tag{3.9}$$

Let $f(X) \coloneqq Ap(X) - \sum_{j\in[Q_x]} B_j p_j(X)$. Notice that because of Eq. (3.9) we have $f(s) = 0$, thus we can assume $f(X) \equiv 0$, as otherwise we can reduce, by Lemma 3.3.2, to the $(Q_x + d + 1)$-DL assumption. It must be the case that both $\sum_{j\in[Q_x]} B_j p_j(s) = 0$ and $A = 0$ because the degree of $p(X)$ and of $p_j(X)$ for any j are different. Moreover, the polynomials $p_j(X)$ are linearly independent, namely the only linear combination $\sum_j a_j p_j(X) = 0$ is the trivial one where the coefficients $a_j = 0$[8], thus $B_j = 0$ for all j. We have that $o''_{w^*,j}\delta_j - o''_{c^*,j} = 0, \forall j$. Thus, we can rewrite the coefficients $o''_{\pi^*,j} = \frac{o''_{c^*,j}}{\delta_j}$, $\forall j$. Since A must be 0:

$$\sum_{i\in[Q_c]} c_{c^*,i}\gamma_i - \sum_{j\in[Q_x]} \frac{o''_{c^*,j}}{\delta_j} - y^* = 0. \tag{3.10}$$

$\mathcal{B}$ can plug the definition of the coefficients $o''_{c^*,j}$ in Eq. (3.10) and derive:

$$\begin{aligned}0 &= \sum_{i\in[Q_c]} c_{c^*,i}\gamma_i - \sum_{i,j} \frac{o'_{c^*,i,j}(\gamma_i - z_{j_i})}{\delta_j} - y^* \\ &= \sum_{i\in[Q_c]} c_{c^*,i}\gamma_i - \sum_i \gamma_i \sum_j \frac{o'_{c^*,i,j}}{\delta_j} + \sum_{i,j} \frac{o'_{c^*,i,j}z_{j_i}}{\delta_j} - y^* \\ &= \sum_{i\in[Q_c]} (c_{c^*,i} - \sum_j \frac{o'_{c^*,i,j}}{\delta_j})\gamma_i + \sum_{i,j} \frac{o'_{c^*,i,j}z_{j_i}}{\delta_j} - y^*.\end{aligned}$$

[8] To see this, $\forall x_j \in \mathcal{Q}_x$ we have that $\sum_{j'} a_{j'}p_{j'}(x_j) = a_j p_j(x_j)$ since $p_j(x_j) \neq 0$ and $p_{j'}(x_j) = 0$ for $j \neq j'$, and $a_j p_j(x_j) = 0$ if and only if $a_j = 0$

Above, the second equation follows from the distributive property of the sum, while in the last step we have grouped the terms depending on γ_i. In particular, the last equation shows that $\mathcal{B}$ can make a forgery in the Aff-MDH game since it knows $z \coloneqq y^* - \sum_{i,j} \frac{o'_{c^*,i,j} z_{j_i}}{\delta_j}$ and coefficients $A_i \coloneqq c_{c^*,i} - \sum_j \frac{o'_{c^*,i,j}}{\delta_j}$ such that:

$$\sum_{i \in [Q_c]} A_i \, [\gamma_i]_1 = [z]_1 \, .$$

For this to be a valid solution in the Aff-MDH game, we need the existence of at least an index i such that $A_i \neq 0$. We show that this occurs with all but negligible probability, i.e., $\Pr[\exists i \in [Q_c] : A_i \neq 0] \geq 1 - \mathsf{negl}(\lambda)$.

To this end, consider an arbitrary $\mu \in [Q_c]$, then we have $\Pr[\forall i \in [Q_c] : A_i = 0] \leq \Pr[A_\mu = 0]$. Thus, for any μ, we have:

$$\Pr[\exists i \in [Q_c] : A_i \neq 0] = 1 - \Pr[\forall i \in [Q_c] : A_i = 0] \geq 1 - \Pr[A_\mu = 0].$$

Below, we argue that $\Pr[A_\mu = 0]$ is negligible based on the randomness of x^* which is chosen by the random oracle after defining A_μ, and we make use of the assumption that $\vec{r}_{\mathsf{c}^*} \neq 0$.

We claim that the value $A_\mu = c_{\mathsf{c}^*,\mu} - \sum_j \frac{o'_{\mathsf{c}^*,j,\mu}}{(x^* - x_j)}$ can be fixed before the random oracle query $\mathbb{x}^*$ is made. To this end, we start by showing that $\vec{o}'_{\mathsf{c}^*,j}$ does not depend on x^*. Let $B(j) \subseteq [Q_c]$ be the subset of indices of the simulation queries that involve x_j and that occurred before the random oracle query that returned x^*. We observe that for every $\eta \in B(j)$ it must be $o_{\mathsf{c}^*,j,\eta} = 0$ since the simulated proof $\pi_{j,\eta}$ is not in the view of the adversary. Therefore:

$$o'_{\mathsf{c}^*,j,i} = \sum_{\eta \in [Q_c]} \vec{R}_{j,\eta,i} \cdot o_{\mathsf{c}^*,j,\eta} = \sum_{\eta \in B(j)} \vec{R}_{j,\eta,i} \cdot o_{\mathsf{c}^*,j,\eta}$$

and observe that all the rows of $\vec{R}_j$ belonging to $B(j)$ can all be defined before x^* is sampled. Hence, we have that A_μ depends on the values $\vec{c}_{\mathsf{c}^*}, x^*, \{x_j\}_j$, and $\vec{o}_{\mathsf{c}^*,j}$ which can all be defined before the random oracle query $\mathbb{x}^*$ is made.

Now, we bound $\Pr[A_\mu = 0]$. Recall that, since the extractor policy Φ_{ext} holds true, we have that $x^* = h(a)$ and $(s, \mathsf{aux}, a) \in \mathcal{Q}_{\mathsf{RO}}$ where $g_c(\mathsf{c}^*, s) = 1$ and the function h is the polynomial encoded in aux_ϕ: the adversary may want to encode up to $n \in \mathsf{poly}(\lambda)$ different polynomials h_i into aux_ϕ to maximize its advantage, and the extractor policy does not impose any restriction on this. Moreover, by the AGM, since $\mathcal{A}_{\mathsf{c}}$ sends a query s (where c^* is encoded in s) to the random oracle it also defines coefficients for c^* before the value a, and therefore $x^* = h(a)$, is defined. Also, it is not hard to see that the representation vector of c^* defined by $\mathcal{A}_{\mathsf{c}}$ when querying the random oracle must be the same representation vector used for the forgery. As otherwise we would break the $(Q_x + d + 1)$-DL assumption. Thus, the coefficients $\vec{c}_{\mathsf{c}^*}$ and $\vec{o}'_{\mathsf{c}^*,j}$ are defined by the adversary before seeing the random value x^*.

Notice that, once the coefficients $\vec{c}_{\mathsf{c}^*}$ and $\vec{o}'_{\mathsf{c}^*,j}$ are fixed, the coefficient A_μ can be seen as function of $x^* \in \mathbb{Z}_q$, i.e. $A_\mu = A_\mu(x^*)$, where:

$$\begin{aligned} A_\mu(X) &\coloneqq c_{\mathsf{c}^*,\mu} + \sum_j \frac{o'_{\mathsf{c}^*,j,\mu}}{X - x_j} \\ &= \frac{c_{\mathsf{c}^*,\mu} \prod_j (X - x_j) + \sum_j (o'_{\mathsf{c}^*,j,\mu} \prod_{j' \neq j} (x_{j'} - X))}{\prod_j X - x_j}. \end{aligned}$$

Notice that $A_\mu(X)(\prod_j(X-x_j))$ vanishes in at most Q_x points in $\mathbb{F}\setminus\mathcal{Q}_x$ and vanishes in the set of points $\mathcal{Q}_x$. Let $\mathcal{R}$ be the set of the roots of such a polynomial, since $\forall i \in [n]$, h_i is defined before x^* is computed, and by union bound:

$$\Pr[\exists i : h_i(\mathsf{RO}(s)) \in \mathcal{R}] \leq \sum_{r\in\mathcal{R}} \Pr[\exists i : h_i(\mathsf{RO}(s)) = r] \leq nQ_x\frac{\max_i \deg(h_i)}{q}$$

for each string s that encodes c^*, To conclude, we notice that $\mathcal{A}$ can submit at most Q_{RO} queries to the RO with strings encoding c^*, say $s_1, \dots s_{Q_{\mathsf{RO}}}$. Thus, the probability that there exist $i \in [n], j \in [Q_{\mathsf{RO}}]$ such that $h_i(\mathsf{RO}(s_j)) \in \mathcal{R}$ is bounded by $nQ_{\mathsf{RO}}Q_x\frac{\max_i \deg(h_i)}{q}$. □

Hybrid $\mathbf{H}_7$. The hybrid $\mathbf{H}_7$ additionally checks that $y^* \neq 0$, and if the condition holds the adversary $\mathcal{A}_{\mathsf{c}}$ loses the game.

Lemma 3.5.7. $\epsilon_6 \leq \epsilon_7 + \epsilon_{(Q_x+1+d)\text{-DL}} + \mathsf{poly}(\lambda)\frac{\deg(h)}{q}$

Proof. We reduce to the evaluation binding of KZG polynomial commitment for polynomials of maximum degree $Q_x + 1 + d$, which, in turn, can be reduced to $(Q_x + 1 + d)$-strong Discrete Log assumption. Let $\mathcal{B}$ be the reduction that upon input $\mathsf{pp}_{\mathbb{G}}, \mathsf{ck} = \left[1, s, \dots, s^{Q_x+d+1}\right]_1, [1, s]_2$ simulates experiment $\mathbf{H}_4$ for the adversary $\mathcal{A}_{\mathsf{c}}$. Eventually, $\mathcal{A}_{\mathsf{c}}$ outputs its forgery $(\mathsf{c}^*, x^*, y^*, \pi^*)$, and $\mathcal{B}$ aborts if $y^* = 0$. The reduction sets $\tilde{f}(X) \coloneqq -y^*p(X)$, sets $y \coloneqq \tilde{f}(x^*)$, and computes π to be a valid KZG proof for $(\left[\tilde{f}(s)\right]_1, x^*, y)$. The forgery against evaluation biding of the reduction is set to be (y, π) and $(0, \pi^*)$ for the commitment $\left[\tilde{f}(s)\right]_1$ on the point x^*. We need to show that:

1. $(\left[\tilde{f}(s)\right]_1, x^*, 0, \pi^*)$ satisfies the verification equation of KZG commitment where the commitment key is set to ck
2. $y \neq 0$

For the first item notice that, by the definition of core adversary, we have that $\vec{r}_{c^*} = \vec{0}$ thus $c^* = 0$. Therefore, by the verification equation:

$$e([0]_1 - y^*\,[p(s)]_1\,, [1]_2) = e(\left[\tilde{f}(s)\right]_1 - 0\,[1]_1\,, [1]_2) = e(\pi^*, [s]_2 - x^*\,[1]_2).$$

For the second item, notice that $\tilde{f}(x^*) = 0$ if and only if x^* is a root of $p(X)$, i.e. $x^* \in \mathcal{Q}_x$ or $x^* = x_r$. Thus, similarly to the previous lemma, by the assumption on the polynomials h_i and by union bound:

$$\Pr[\exists i : h_i(\mathsf{RO}(s)) \in \mathcal{Q}_x \cup \{x_r\}] \leq nQ_{\mathsf{RO}}(Q_x + 1)\frac{\max_i \deg(h_i)}{q}.$$

□

Finally, we have that the probability that the adversary wins in $\mathbf{H}_7$ is null, namely $\epsilon_7 = 0$. Indeed, the canonical extractor $\mathcal{E}$ outputs the 0 polynomial, moreover because of the condition introduced in $\mathbf{H}_6$, we have $\mathsf{c}^* = [0]_1$, and because of the condition introduced in $\mathbf{H}_7$ we have $y^* = 0$, thus the witness extracted is valid for the instance $\mathbb{x}^* = (\mathsf{c}^* = [0]_1\,, x^*, y^* = 0)$. □

3.5.2 Simulation extractability of Hiding KZG

We extend the result of Theorem 3.5.1 to the case in which the $\mathsf{CP}_{\mathtt{evl}}$ based on KZG uses hiding commitments. In particular, we show a reduction to the simulation extractability of non-hiding KZG for uniform distributions $\mathcal{U}_\ell$ of the commitments.

Theorem 3.5.2. *Let $\mathcal{U}_\ell$ be the distribution that outputs ℓ uniformly random group elements in $\mathbb{G}_1$. $\forall \ell \in \mathbb{N}$, the hiding KZG $\mathsf{CP}_{\mathtt{evl}}$ is $\Phi_{\mathcal{U}_\ell}$-simulation-extractable in the AGM. In particular, there exists $\mathcal{E}$ such that for any algebraic adversary $\mathcal{A}$:*

$$\mathbf{Adv}^{\Phi\text{-}se}_{\mathsf{CP}_{\mathtt{evl}},\mathcal{A},\mathcal{S},\mathcal{E},\mathcal{U}_\ell}(\lambda) \leq O(\ell \epsilon_{q\text{-}\mathsf{DL}}(\lambda)) + \mathsf{poly}(\lambda)\epsilon_h(\lambda)$$

where d is the maximum degree supported by $\mathsf{CP}_{\mathtt{evl}}$, $\epsilon_{(Q_x+d+1)\text{-}\mathsf{DL}}(\lambda)$ is the maximum advantage for any algebraic PT adversary against the (Q_x+d+1)-strong Discrete-Log Assumption, $\epsilon_{\mathcal{D}\text{-}\mathsf{Aff}\text{-}\mathsf{MDH}}(\lambda)$ is the maximum advantage for any algebraic PT adversary against the $\mathcal{D}$-Aff-MDH Assumption, h is the polynomial that satisfies the Polynomial check of $\Phi_{\mathcal{D}}$, and $\epsilon_h = \frac{\mathtt{deg}(h)}{q}$.

Proof. We reduce an adversary $\mathcal{A}$ for the simulation extractability of $\Pi_{\mathtt{evl}}$ with hiding commitments to an adversary $\mathcal{B}$ for the simulation extractability with non-hiding commitments and a uniform distribution that generates twice as many commitments. $\mathcal{B}$ receives as input the description of a bilinear group $\mathsf{pp}_{\mathbb{G}}$ and an SRS $\mathsf{srs} := (\mathsf{ek}, \mathsf{vk})$ where $\mathsf{ek} = \left[1, s, \ldots, s^d\right]_1$ and $\mathsf{vk} = [1, s]_2$; $\mathcal{B}$ samples $\alpha \leftarrow\!\!\$\ \mathbb{Z}_q$ and computes $\mathsf{ek}_\alpha \leftarrow \left[\alpha, \alpha s, \ldots, \alpha s^d\right]_1$ and forwards to $\mathcal{A}$ the SRS $\mathsf{srs}' := ((\mathsf{ek}, \mathsf{ek}_\alpha), \mathsf{vk})$. Finally, $\mathcal{B}$ parses the list of commitments coms as two lists of equal size $(\mathsf{coms}_1, \mathsf{coms}_2)$ and gives $\mathcal{A}$ the commitments $\mathsf{coms}' = \mathsf{coms}_1 + \alpha\mathsf{coms}_2$. Every time $\mathcal{A}$ comes up with some group element c, attaches a representation vector $\vec{r}_c$ that we can parse as $\vec{f}_c \| \vec{v}_c \| \vec{c}_c \| \vec{o}_c$ where $\vec{f}_c$ (resp. $\vec{v}_c$) is the vector of coefficients associated to group elements ek (resp. ek_α), $\vec{c}_c$ is the vector of coefficients associated to group elements coms', and $\vec{o}_c$ is the vector of coefficients associated to the group elements of the simulated proofs proofs'. The strategy of $\mathcal{B}$ is to forward the simulation queries of $\mathcal{A}$ whose representation vectors are given w.r.t. srs', attaching the representation vectors w.r.t. srs.

On input a simulation query of the form $\mathtt{x} = (\mathsf{c}, x_j, y)$ and leakage y', with $\vec{r}_c = \vec{f}_c \| \vec{v}_c \| \vec{c}_c$ as representation vector for c, $\mathcal{B}$ invokes the simulation oracle $\mathcal{S}_1$ first on input $\mathtt{x}_1 = (\langle \vec{f}_c + \alpha\vec{v}_c, \mathsf{ek}\rangle + \langle \vec{c}_c \| \vec{0}, \mathsf{coms}\rangle, x_j, y)$ and then on $\mathtt{x}_2 = (\langle \vec{0} \| \vec{c}_c, \mathsf{coms}\rangle, x_j, y')$, obtaining the two proofs π_1 and π_2. Finally returns to $\mathcal{A}$ the proof $\pi := \pi_1 + \alpha\pi_2$ which satisfies the verification equation:

$$e(\mathsf{c} - [y]_1 - [\alpha y']_1, [1]_2) = e(\pi, [s - x_j]_2)$$

and adds π to proofs'. When $\mathcal{A}$ makes the forgery $(\mathsf{c}^*, x^*, y^*, y'^*, \pi^*)$, with associated representation vectors $\vec{r}'_\tau = \vec{f}_\tau \| \vec{v}_\tau \| \vec{c}_\tau \| \vec{o}_\tau$, for $\tau \in \{c^*, \pi\}$, $\mathcal{B}$ forwards the forgery $(\mathsf{c}, x^*, y^* + \alpha y'^*, \pi^*)$. $\mathcal{B}$ needs to attach representation vectors for the group elements c^*, π^* w.r.t. $\mathsf{srs}, \mathsf{coms}$ and proofs; to do that, $\mathcal{B}$ applies the following transformation and computes $\vec{r}_\tau = \vec{f}_\tau + \alpha\vec{v}_\tau \| \vec{c}_\tau \| \alpha\vec{c}_\tau \| (o_{\tau,i} \| \alpha o_{\tau,i})_i$. This choice, indeed, is such that:

$$\langle \vec{r}'_\tau, (\mathsf{ek} \| \mathsf{ek}_\alpha \| \mathsf{coms}' \| \mathsf{proofs}')\rangle = \langle \vec{r}_\tau, (\mathsf{ek} \| \mathsf{coms} \| \mathsf{proofs})\rangle$$

If the forgery of $\mathcal{A}$ is valid, so is the forgery of $\mathcal{B}$ and, unless with negligible probability, the canonical extractor $\mathcal{E}$ extracts the valid witness $f(X) + \alpha v(X)$. □

3.5.3 Simulation extractability of batch KZG

Here we generalize the scheme described in Section 3.5.1 to the batched setting, highlighting the parts of the construction that are not needed if hiding is not desired. In particular, this scheme $\mathsf{CPm\text{-}evl}$ allows one to prove that for all i the polynomial f_i committed in c_i evaluates to y_i on the point x. This batched version, which is given in the ROM, follows from [GWC19, MBKM19] and relies on the linearity of the polynomials and the homomorphic properties of KZG. Our contribution is to prove its policy-based simulation extractability.

$\mathsf{KGen}_{\mathtt{m\text{-}evl}}$: parse ck as $(([s^j]_1)_{j\in[0,d]}, ([\alpha s^j]_1)_{j\in[0,d]}, [1,s]_2)$ and define $\mathsf{ek} := \mathsf{ck}$ and $\mathsf{vk} := [1,s]_2$, and return $\mathsf{srs} := (\mathsf{ek}, \mathsf{vk})$.

$\mathsf{Prove}_{\mathtt{m\text{-}evl}}(\mathsf{ek}, \mathbb{x} = (x, (\mathsf{c}_i, y_i)_i), \mathbb{w} = (f_i, r_i)_i)$: for all i compute the polynomials $\pi_i(X)$ such that $\pi_i(X)(X-x) \equiv f_i(X) - y_i$, the polynomials $\pi'_i(X)$ such that $\pi'_i(X)(X-x) \equiv r_i(X) - r(x)$, $\rho \leftarrow \mathsf{RO}(\mathsf{batch}\|\mathbb{x})$, and output

$$\left(\sum_i \rho^{i-1} \left[\pi_i(s) + \alpha\pi'(s)\right]_1 , \sum_i \rho^{i-1} r_i(x) \right)$$

$\mathsf{Verify}_{\mathtt{m\text{-}evl}}(\mathsf{vk}, \mathbb{x} = (x, (\mathsf{c}_i, y_i)_i), (\pi, y'))$: compute $\rho \leftarrow \mathsf{RO}(\mathsf{batch}\|\mathbb{x})$ and output 1 if and only if

$$e(\sum_i \rho^{i-1}\mathsf{c}_i - \left[\sum_i \rho^{i-1} y_i\right]_1 - [\alpha y']_1, [1]_2) = e(\pi, [s-x]_2).$$

Similarly to CPevl, we can prove that $\mathsf{CPm\text{-}evl}$ achieves F-leaky zero-knowledge (see Definition 3.4.3) where $F(\mathbb{x} = (x, (\mathsf{c}_i, y_i)_i), \mathbb{w} = (f_i, r_i)_i) = \sum_i \rho^{i-1} r_i(x)$ and $\rho = \mathsf{RO}(\mathsf{batch}\|\mathbb{x})$.

We define the simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1, \mathcal{S}_2)$, where $\mathcal{S}_0$ outputs the trapdoor information s, α together with the srs, $\mathcal{S}_2$ simulates the random oracle on any input via lazy sampling, and $\mathcal{S}_1$ simulates proofs for $\mathbb{x} = (x, (\mathsf{c}_i, y_i)_i)$ and leakage y' by outputting $\pi = ((\mathsf{c} - [y]_1 - [\alpha y']_1)(s-x)^{-1}, y')$, where $\mathsf{c} \leftarrow \sum_i \rho^{i-1}\mathsf{c}_i$, $y \leftarrow \sum_i \rho^{i-1} y_i$ for $\rho \leftarrow \mathcal{S}_2(\mathsf{batch}\|\mathbb{x})$.

The extraction policy. The extraction policy $\mathbf{\Phi}^{\mathtt{s\text{-}adpt}}_{\mathtt{m\text{-}evl}}$ is naturally extended from the single point case of CPevl. The evaluation points x_j for the instances for which the adversary can see simulated proofs are selectively chosen independently of the commitment key, while the evaluation values $(y_i)_i$ can be arbitrarily chosen by the adversary. Each policy $\Phi_{\mathcal{D}}$ in the family is a tuple of the form $(\Phi_0^{\mathcal{D}}, \Phi_1)$. Each policy $\Phi_{\mathcal{D}}$ is a tuple of the form $(\Phi_0^{\mathcal{D}}, \Phi_1)$, as defined in Section 3.4.1, where $\Phi_0^{\mathcal{D}}$ outputs the parameters pp_Φ while Φ_1 outputs a verdict bit. In particular, $\Phi_0^{\mathcal{D}}$ on input group parameters $\mathsf{pp}_{\mathbb{G}}$ outputs $\mathsf{pp}_\Phi := (\mathsf{coms}, \mathcal{Q}_x)$, where coms is a vector of commitments sampled from $\mathcal{D}$, and $\mathcal{Q}_x$ is a set of $\mathcal{Q}_x$ evaluation points.

For sake of clarity, we define the policy Φ_1 as the logical conjunction of a "simulator" policy Φ_{sim} and an "extractor" policy Φ_{ext}, i.e. $\Phi_1 = \Phi_{\mathsf{sim}} \wedge \Phi_{\mathsf{ext}}$, the first defines rules under which we can classify a simulation query *legal*, while the second defines rules under which the extractor must be able to extract a meaningful witness. We highlight the parts needed only for the hiding setting.

Definition 3.5.4. *Let Φ_{sim} be the policy that returns 1 if and only if:*

1. **Points check:** *let $(\mathbb{x}_i, \mathsf{aux}_i, \pi_i)_i$ be all the entries of $\mathcal{Q}_{\mathsf{sim}}$. Recall that an instance $\mathbb{x}$ can be parsed as $(x, (\vec{c}, \vec{y}))$, check that $\forall i : \mathbb{x}_i.x \in \mathcal{Q}_x$.*

2. **Commitment Check:** *For any $i \in [Q_{\mathsf{sim}}]$, parse aux_i as the leakage value y'_i and the representation vectors for $\mathbb{x}_i.\vec{c}$ and π_i; in particular, let $\vec{M}_i = \vec{F}_i \| \vec{V}_i \| \vec{C}_i$ be the algebraic representation of the commitments $\mathbb{x}_i.\vec{c}$. For any i check that $\left(\vec{F}_i \| \vec{V}_i\right) \cdot \mathsf{ek} + \vec{C}_i \cdot \mathsf{coms} = \mathbb{x}_i.\vec{c}$ (namely, that the commitments in the instance $\mathbb{x}_i$ are computed as a linear combination of the simulated commitments and the elements of $\mathbb{G}_1$ of the SRS)*

3. **Algebraic Consistency:** *The simulation queries satisfy the algebraic consistency for $\mathsf{CP}_{\mathtt{m\text{-}evl}}$. Namely, let $\mathcal{I}_J = \{i : \mathbb{x}_i.x = x_j\}$ and let $\vec{R}_j = (\vec{C}_i)_{i \in \mathcal{I}_J}$. Check that for all j: (i) the system of linear equations $\vec{R}_j \cdot \vec{z} = \vec{y}_j$ has at least a solution, where $\vec{z}$ are the variables and $\vec{y}_j = (\mathbb{x}_i.\vec{y} + \vec{F}_i \cdot (1, x_j, \ldots, x_j^d)^\top)_{i \in \mathcal{I}_J}$, and (ii) the system of linear equations $\vec{R}_j \cdot \vec{z}' = \vec{y}'_j$ has at least a solution, where $\vec{z}'$ are the variables and $\vec{y}'_j = (y'_i + (1, \rho_i, \ldots, \rho_i^m) \cdot \vec{V}_i \cdot (1, x_j, \ldots, x_j^d)^\top)_{i \in \mathcal{I}_J}$, and $\rho_i = \mathsf{RO}(\mathsf{batch} \| \mathbb{x}_i)$.*

Next, we define the policy Φ_{ext} as the logical disjunction of two policies. We recall and extend the notation introduced in Section 3.5.1: let $g_c \colon \mathbb{G}_1^* \times \{0,1\}^* \to \{0,1\}$ be a function that on inputs a list of group elements c_i and a string s, that can be parsed as a list of group elements $\hat{\mathsf{c}}_i$ followed by a second string $\tilde{s}$, outputs 1 if and only if $\forall i, \exists j : \mathsf{c}_i = \hat{\mathsf{c}}_j$.

Definition 3.5.5. *Let $\Phi_{\mathsf{ext}}, \Phi^{rnd}_{\mathsf{ext}}$ and $\Phi^{der}_{\mathsf{ext}}$ be predicates that parse the forgery instance $\mathbb{x}^* = (x^*, \vec{c}^*, \vec{y}^*)$.*

- *$\Phi^{rnd}_{\mathsf{ext}}$ returns 1 if and only if there exists a query (s, aux, a) to the random oracle and aux contains a non-constant polynomial $h(X)$ such that the following conditions are satisfied:*
  1. **Hashing check:** $(s, \mathsf{aux}, a) \in \mathcal{Q}_{\mathsf{RO}}$
  2. **Decoding check:** $g_c(\vec{c}^*, s) = 1$.
  3. **Polynomial check:** $g_h(h, \mathsf{aux}) = 1$, *where $g_h \colon \mathbb{F}[X] \times \{0,1\}^* \to \{0,1\}$ is a function that on input a polynomial $h(X)$ and a string aux outputs 1 if and only if $h(X)$ is encoded in aux.*
  4. **Computation check:** $h(a) = x^*$.
- *$\Phi^{der}_{\mathsf{ext}}$ returns 1 if and only if $\exists (\mathbb{x}, \cdot, \pi) \in \mathcal{Q}_{\mathsf{sim}}$ s.t. $\mathbb{x} := (x^*, \vec{c}^*, \vec{y}')$ and $(\vec{y}', \pi) \neq (\vec{y}^*, \pi^*)$.*
- *Φ_{ext} returns the logical disjunction of $\Phi^{rnd}_{\mathsf{ext}}$ and $\Phi^{der}_{\mathsf{ext}}$.*

Theorem 3.5.3. *For any witness sampleable distribution $\mathcal{D}$ that is $\mathcal{D}$-Aff-MDH-secure (see Definition 3.3.1), any bilinear-group generator $\mathsf{GroupGen}$ that samples the generator of the group $\mathbb{G}_1$ uniformly at random, $\forall \Phi_{\mathcal{D}} \in \mathbf{\Phi}^{\mathtt{s\text{-}adpt}}_{\mathtt{m\text{-}evl}}$, the scheme $\mathsf{CP}_{\mathtt{m\text{-}evl}}$ is $\Phi_{\mathcal{D}}$-simulation-extractable in the AGM. In particular, there exists $\mathcal{E}$ such that for any algebraic adversary $\mathcal{A}$:*

$$\mathbf{Adv}^{\Phi_{\mathcal{D}}\text{-}\mathsf{se}}_{\mathsf{CP}_{\mathtt{m\text{-}evl}},\mathcal{A},\mathcal{S},\mathcal{E}}(\lambda) \leq O(\epsilon_{(Q_x+d+1)\text{-}\mathsf{DL}}(\lambda)) + O(\epsilon_{\mathit{Aff\text{-}MDH}}(\lambda)) + \mathsf{poly}(\lambda)\epsilon_h$$

where d is the maximum degree supported by $\mathsf{CP}_{\mathtt{m\text{-}evl}}$, $\epsilon_{(Q_x+d+1)\text{-}\mathsf{DL}}(\lambda)$ is the maximum advantage for any algebraic PT adversary against the $(Q_x + d + 1)$-strong Discrete-Log Assumption, $\epsilon_{\mathit{Aff\text{-}MDH}}(\lambda)$ is the maximum advantage for any algebraic PT adversary against the $\mathcal{D}$-Aff-MDH Assumption, h is the polynomial that satisfies the Polynomial check of $\Phi_{\mathcal{D}}$, and $\epsilon_h = \frac{\deg(h)}{q}$.

3.6 Simulation-Extractable Universal zkSNARKs

In this section we show our compiler for universal SNARKs based on polynomial IOPs.

In our work, we use PIOPs with some slight refinements.[9] The first one, called *(non-adaptive) algebraic verifiers* (see Definition 3.6.1), says that the above polynomials $v_j^{(k)}$ do not depend on the instance and can be expressed as polynomial functions of $\mathcal{V}$'s random coins, i.e., $v_j^{(k)}(X) = \tilde{v}_j^{(k)}(X, \vec{\rho})$ for some instance-independent $\tilde{v}_j^{(k)}$.

Definition 3.6.1 (Non-adaptive Algebraic Verifier)**.** *A PIOP* PIOP *is* (non-adaptive) algebraic verifier *if there exists an alternative deterministic PT algorithm* $\tilde{\mathcal{V}}$ *such for any* $\mathbb{i}$ *and* $\mathbb{x}$ *we have* $(\tilde{v}_1^{(k)}, \ldots, \tilde{v}_n^{(k)})_{k\in[n_e]} \leftarrow \tilde{\mathcal{V}}(\mathbb{F}, |\mathbb{i}|)$ *where for any* $j \in [n]$ *and* $k \in [n_e]$ *we have* $\tilde{v}_j^{(k)} \in \mathbb{F}[X, X_1, \ldots, X_{r-1}]$*. Also, for any field elements* $\rho_1, \ldots, \rho_{r-1}$*, let* $(G^{(k)}, v_1^{(k)}, \ldots, v_n^{(k)})_{k\in[n_e]} \leftarrow \mathcal{V}(\mathbb{F}, \mathbb{i}, \mathbb{x}, \rho_1, \ldots, \rho_{r-1})$*, we have for any* $i \in [r(|\mathbb{i}|) - 1], j \in [n_{i-1}, n_i], k \in [n_e]$*:*

$$\tilde{v}_j^{(k)}(X, \rho_1, \ldots, \rho_{r-1}) = v_j^{(k)}(X)$$

The second one is a more restrictive[10] concept of soundness called *state-restoration straight-line knowledge soundness* (see Definition 3.6.2). This combines the notion of state-restoration soundness from [BCS16] with the concept of straight-line extractability from [CFF+21]. For further clarification, the malicious prover $\tilde{\mathsf{P}}$ engages in a game with the honest verifier V and has the additional ability to *roll back* the interaction with the verifier to a previous state. At some point, the interaction may reach a final state. The prover is considered successful if it produces an accepting transcript, while the extractor, given such a transcript that includes all the oracle polynomials, fails to produce a valid witness.

Definition 3.6.2 (State-restoration (straight-line) proof of knowledge)**.** *Let* $\mathbf{Exp}^{sr}_{\tilde{\mathsf{P}},\mathsf{PIOP},\mathcal{E}}(\mathbb{F})$ *be the experiment in Fig. 3.2. A PIOP* PIOP *is* state-restoration (straight-line) proof of knowledge *if there exists an extractor* $\mathcal{E}$ *such that for any* $\tilde{\mathsf{P}}$ *and any* $\mathbb{F}$*:*

$$\Pr\left[\mathbf{Exp}^{sr}_{\tilde{\mathsf{P}},\mathsf{PIOP},\mathcal{E}}(\mathbb{F}) = 1\right] \leq \mathsf{negl}(|\mathbb{F}|)$$

The third one is that we do not explicitly check the degree of the polynomials. This is mainly for simplicity in the presentation of the compiler since, for each poly p_i where we check degree d_i the prover P additionally sends $p'_i(X) = p(X) \cdot X^{D-d_i}$ where D is the maximum degree, and V additionally checks the equation $\sum \rho^i \cdot (X^{D-d_i} p_i(X) - p'_i(X)) \equiv 0$, for a randomizer ρ.

Similarly to previous work, we use the notion of bounded zero-knowledge of [CHM+20, CFF+21].

A list $\mathcal{L} = \{(i_1, y_1), \ldots\}$ is $(\vec{b}, C)$-bounded where $\vec{b} \in \mathbb{N}^n$ and C is a PT algorithm if $\forall i \in [n] : |\{(i, y) : (i, y) \in \mathcal{L}\}| \leq \vec{b}_i$ and $\forall (i, y) \in \mathcal{L} : C(i, y) = 1$.

Definition 3.6.3 ($\vec{b}$-bounded Zero-Knowledge)**.** *A PIOP* PIOP *is* $\vec{b}$*-Zero-Knowledge if there exists a checker* C *such that* $\Pr[C(i, x) = 0] \leq \mathsf{negl}(|\mathbb{F}|)$ *over random* x *such that for every index* $\mathbb{i}$*, and* $(\mathsf{pp}, \mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$*, and every* $(\vec{b}, C)$*-bounded list* $\mathcal{L}$*, the following random variables are within* ϵ *statistical distance:*

$$\left(\mathsf{view}\left(\mathsf{P}(\mathbb{F}, \mathbb{i}, \mathbb{x}, \mathbb{w}), \mathcal{V}^{\mathsf{I}(\mathbb{F},\mathbb{i})}(\mathbb{F}, \mathbb{x})\right), (p_i(y))_{(i,y)\in\mathcal{L}}\right) \approx_\epsilon \mathcal{S}(\mathbb{F}, \mathbb{i}, \mathbb{x}, \mathcal{L})$$

[9] All the PIOPs that we are aware of satisfy both these properties.

[10] The (classical) notion of knowledge extractability implies state-restoration soundness through complexity leveraging [BCS16].

1. The challenger initializes the list $\mathsf{SeenStates}$ to be empty.
2. Repeat the following until the challenger halts:
   (a) $\tilde{\mathsf{P}}$ either (1) chooses a complete verifier state cvs in $\mathsf{SeenStates}$ or (2) sends a fresh tuple $(\mathbb{i}, \mathbb{x}, \{\pi_{1,j}\}_j, \{p_{1,j}\}_j)$ to the challenger.
   (b) If (1) the challenger sets the verifier to cvs:
      i. if $\mathsf{cvs} = (\mathbb{i}, \mathbb{x}, \{\pi_{1,j}\}_j, \{p_{1,j}\}_j \| \rho_1 \| \ldots \| \{\pi_{i,j}\}_j, \{p_{i,j}\}_j)$ and $i < r(\mathbb{x})$: $\tilde{\mathsf{P}}$ outputs $\{\pi_{i-1,j}\}_j$, $\{p_{i-1,j}\}_j$; V samples ρ_i and sends it to $\tilde{\mathsf{P}}$; the game appends $\mathsf{cvs}' := (\mathsf{cvs} \| \{\pi_{i-1,j}\}_j \| \{p_{i-1,j}\}_j \| \rho_i)$ to the list $\mathsf{SeenStates}$;
      ii. if $\mathsf{cvs} = (\mathbb{i}, \mathbb{x}, \{\pi_1, j\}_j, \{p_1, j\}_j \| \rho_1 \| \ldots \| \rho_{r-1})$: $\tilde{\mathsf{P}}$ outputs $\{\pi_{r,j}\}_j$ and $\{p_{r,j}\}_j$; the challenger runs $\mathsf{I}(\mathbb{F}, \mathbb{i})$ and V performs the decision phase of the PIOP. The challenger sets cvs to be the final cvs, sets the decision bit d as the output of the verifier V and halts.
   (c) If (2) the verifier samples ρ_1 and sends it to $\tilde{\mathsf{P}}$; the game appends the state $\mathsf{cvs}' := (\mathbb{i}, \mathbb{x}, \{\pi_{1,j}\}_j, \{p_{1,j}\}_j \| \rho_1)$ to the list $\mathsf{SeenStates}$.
3. The game computes the extraction bit $b \stackrel{\text{def}}{=} (\mathbb{i}, \mathbb{x}, \mathcal{E}(\mathbb{i}, \mathbb{x}, p_1, \ldots, p_n)) \in \mathcal{R}$ where the instance $\mathbb{x}$ and the polynomials $p_1, \ldots, p_n$ are the ones generated by I and the ones included in the final cvs. The game returns $(d \wedge \neg b)$, i.e., the malicious prover convinces the verifier but the extractor fails.

Figure 3.2: The $\mathbf{Exp}^{sr}_{\tilde{\mathsf{P}},\mathsf{PIOP},\mathcal{E}}(\mathbb{F})$ experiment.

where $p_1, \ldots, p_n$ are the polynomials returned by the prover P and $\mathcal{S}$ is a PPT simulator. Moreover, PIOP *is* special honest-verifier $\vec{b}$-bounded zero-knowledge *if $\mathcal{S}$ first samples uniformly at random the verifier's messages $\rho_1, \ldots, \rho_{r-1}$ and then computes the full transcript. Namely, $\mathcal{S}$ can take as additional input the verifier's messages.* PIOP *is* independent leakage *if $\mathcal{S}$ can be divided in two algorithms $(\mathcal{S}_0, \mathcal{S}_1)$ where $\mathcal{S}_0$ outputs the simulated transcript, and $\mathcal{S}_1$ the leakage. Namely, $\mathcal{S}(\mathbb{F}, \mathbb{i}, \mathbb{x}, \mathcal{L}; r) = \mathcal{S}_0(\mathbb{F}, \mathbb{i}, \mathbb{x}; r), \mathcal{S}_1(\mathbb{F}, \mathbb{i}, \mathbb{x}, \mathcal{L}; r)$.*

Compilation-safe PIOP. We must incorporate an additional element to the classical recipe. As stated in the introduction, mix-and-match attacks on compiled protocols, involving two or more independent sub-protocols, are unavoidable. Therefore, we identify a structural restriction on the PIOP that prevents such problematic scenarios. The restriction is easy to state and easy to meet:

Definition 3.6.4 (Compiler-safe PIOP)**.** *A PIOP* PIOP *is* compiler-safe *if for any $\mathbb{i}, \mathbb{x}$ and $\vec{\rho} := \rho_1, \ldots, \rho_{r-1}$ and any tuple $(G^{(k)}, v_1^{(k)}, \ldots, v_n^{(k)})_{j \in [n_e]} \leftarrow \mathcal{V}(\mathbb{F}, \mathbb{i}, \mathbb{x}, \vec{\rho})$ there exists an index k such that for all j the polynomials $v_j^{(k)}$ are of degree at least one.*

We notice that some PIOPs have this technical condition (an example is [GWC19]) while other PIOPs that internally run different sub-protocols might not have this property. However,

at PIOP level we can always obtain this property by adding an extra degree of randomness to the polynomials (obtaining $\vec{b}+\vec{1}$-bounded zero-knowledge) and adding an extra trivial equation over the polynomials with the v_j set to be the identity function.

Commitments Simulator. We abstract how our compiler handles the simulation of the commitments for the oracles sent during a PIOP protocol's execution.

Definition 3.6.5 (Commitment Simulator for PIOP)**.** *Let* PIOP *be a PIOP with* $\vec{b}$*-Zero-Knowledge simulator* $\mathcal{S}$ *and checker* C *and consider a two-stage PPT algorithm where* $\mathcal{S}_{\mathsf{Com}}(0,\mathsf{ck})$ *outputs a vector* coms *of commitments, and* $\mathcal{S}_{\mathsf{Com}}(1,|\mathbb{i}|,\mathbb{x})$ *outputs a matrix* $\vec{M}_{\mathbb{i},\mathbb{x}}$ *of linearly independent rows.* $\mathcal{S}_{\mathsf{Com}}$ *is a* commitment simulator *for a* CP *(defined over the same commitment scheme) and* PIOP *if for every* $(\vec{b},C)$*-bounded list* $\mathcal{L}$*,* $\mathbb{F},\mathbb{i}$ *and* $\mathbb{x}$ *the following distributions are computationally indistinguishable:*

$$\Big(\mathsf{ck},\mathsf{view}\Big(\mathsf{P}(\mathbb{F},\mathbb{i},\mathbb{x},\mathbb{w}),\mathcal{V}\Big),(p_j(x))_{(j,x)\in\mathcal{L}},(\mathsf{Com}(\mathsf{ck},p_i))_{i\in[n]}\Big)\approx_c$$
$$\Big(\mathsf{ck},\mathcal{S}(\mathbb{F},\mathbb{i},\mathbb{x},\mathcal{L}),(\vec{M}_{\mathbb{i},\mathbb{x}}\cdot\mathsf{coms}\|\mathsf{ck})\Big)$$

Moreover, we have that the view $(\mathcal{L},\mathcal{S}(\mathbb{F},\mathbb{i},\mathbb{x},\mathcal{L}),\mathbb{x},\vec{M}_{\mathbb{i},\mathbb{x}})$ *satisfies the algebraic consistency for the CP-SNARK* CP *(as in Definition 3.5.1).*

The definition is in two stages so to fit our result of Theorem 3.5.1. For KZG the first part handles the generation of instance-independent commitments according to a $\mathcal{D}_k$-Aff-MDH Assumption, while the second part acts over the formerly sampled commitments adapting them to the instance at hand. If the commitment scheme is hiding then the task of defining the commitment simulator $\mathcal{S}_{\mathsf{Com}}$ for an arbitrary PIOP is trivial (the distribution of $\mathcal{S}_{\mathsf{Com}}(0,\mathsf{ck})$ is uniformly over the commitment space), thus this property is interesting for the case of non-hiding commitments. Similarly, this notion can be applied to commitment schemes that are not homomorphic: in that case, we must require $\vec{M}_{\mathbb{i},\mathbb{x}}$ to be the identity.

3.6.1 The Compilation-Ready CP-SNARK

Instead of compiling directly a PIOP through a polynomial commitment in its simplest form (i.e., an evaluation proof for each polynomial queried in the PIOP), we take an alternative road similar to [CFF+21]. Namely, we assume the existence of a CP-SNARK that, w.r.t. a tuple of commitments $(\mathsf{c}_j)_{j\in[n]}$, is capable of proving either knowledge of polynomials $(p_j)_{j\in[n]}$ opening these commitments, or that the committed polynomials satisfy a statement like the one in Eq. (2.2) (i.e., that the oracles committed in $(\mathsf{c}_j)_{j\in[n]}$ would make the PIOP verifier accept)[11]. We call this building block a *compilation-ready* CP-SNARK (CP, shortly), and informally we refer to the former type of statements as "proof of knowledge" and to the latter as "PIOP verifier". While our compilation strategy follows previous work, our novel contribution is to properly define the properties that this CP-SNARK must satisfy in order to argue that the result of the compiler is simulation-extractable, and not only knowledge-sound. These properties are mainly three. The first one is that the CP prover can "append" arbitrary messages to the proven instances. Looking ahead to our compiler, this feature is used so that prover and

[11]The reason to assume a single CP-SNARK for both kind of statements, instead of one for polynomial equations and one for openings, has to do with the security guarantees when we compose protocols in the AGM [ABK+21].

verifier can append the (hash of the) protocol's transcript to the proven instance, in such a way that a CP proof acts as a *signature of knowledge for the transcript*. Note, this hashing of the transcript already happens in the standard PIOP compiler due to the application of the Fiat-Shamir transform; here, we highlight it explicitly as it plays an important role in the proof of simulation extractability. The second property, referred to as the *commitment simulator for PIOP* (see Definition 3.6.5), intuitively requires the existence of a strategy to simulate commitments such that: adding them to the view preserves zero-knowledge, and the simulation respects the "commitment check" constraint in Item 2 of Definition 3.5.2. This is a very mild property that is trivially satisfied when employing hiding commitments, and is met by existing simulation strategies based on deterministic commitments to randomized polynomials [GWC19, CFF+21]. The third property of CP is that it must be *simulation-extractable* w.r.t. a policy $\hat{\mathbf{\Phi}}$ such that:

- The adversary can ask simulated proofs for "PIOP verifier" statements where all the $v_j^{(k)}$ of Eq. (2.2) are fixed at the beginning of the experiment.
- If the forgery of the adversary is a "proof of knowledge" for commitments $\vec{\mathsf{c}}^*$, then the adversary must return as auxiliary output yet another forgery for a "PIOP verifier" statement such that: (1) All the commitments $\vec{\mathsf{c}}^*$ appear in the second forgery, (2) the second forgery is valid according to the extractor policy described next.
- If the forgery of the adversary is for the "PIOP verifier" statement, then the statement-proof pair returned by the adversary must not be in the list of simulated statements-proofs, and (similarly to Definition 3.6.4) there exists a k such that for all j the polynomial $v_j^{(k)}$ has degree at least 1.

Let $\mathcal{R}_{\mathtt{poly}}$ be the relation that upon relation parameters the commitment key ck and an instance $\mathbb{x}_{\mathtt{poly}} := \left(\mathtt{msg}, \vec{\mathsf{c}}, (\bar{G}^{(k)}, \vec{v}^{(k)})_{k\in[n_e]}\right)$, where $\mathtt{msg}$ is an arbitrary string and $\vec{\mathsf{c}} := (\mathsf{c}_j)_{j\in[n]}$, and whose witness is $\mathbb{w}_{\mathtt{poly}} := (\vec{p}, \vec{o})$, outputs 1 if and only if:

$$\begin{aligned}&\forall k \in [n_e]: \ \bar{G}^{(k)}(X, p_1(v_1^{(k)}(X)), \ldots, p_n(v_n^{(k)}(X))) \equiv 0 \quad \wedge \\ &\forall j : \mathsf{VerCom}(\mathsf{ck}, \mathsf{c}_j, p_j, o_j) = 1\end{aligned}$$

Notice that a simulation-extractable CP-SNARK for the relation $\mathcal{R}_{\mathtt{poly}}$ forms a *signature of knowledge* [CL06] for an instance $\left(\vec{\mathsf{c}}, (\bar{G}^{(k)}, \vec{v}^{(k)})_{k\in[n_e]}\right)$ and the message $\mathtt{msg}$. Consider the relation for multi-instance opening of commitments defined below:

$$\mathcal{R}_{\mathtt{opn}}(\mathsf{ck}, \mathbb{x}_{\mathtt{opn}} = \vec{\mathsf{c}}, \mathbb{w}_{\mathtt{opn}} = (\vec{p}, \vec{o})) := (\forall j \in [|\vec{\mathsf{c}}|] : \mathsf{VerCom}(\mathsf{ck}, \mathsf{c}_j, p_j, o_j)) \,.$$

Our compiler needs a CP-SNARK that can simultaneously prove the polynomial evaluation relation and the knowledge of the openings. Let $\hat{\mathcal{R}}$ be the joint relation of $\mathcal{R}_{\mathtt{poly}}$ and $\mathcal{R}_{\mathtt{opn}}$, i.e.:

$$\hat{\mathcal{R}} \stackrel{\text{def}}{=} \{\mathsf{ck}, (\mathtt{poly}, \mathbb{x}), \mathbb{w} : \mathcal{R}_{\mathtt{poly}}(\mathsf{ck}, \mathbb{x}, \mathbb{w})\} \cup \{\mathsf{ck}, (\mathtt{opn}, \vec{\mathsf{c}}), \mathbb{w} : \mathcal{R}_{\mathtt{opn}}(\mathsf{ck}, \vec{\mathsf{c}}, \mathbb{w})\} \tag{3.11}$$

We say that a statement of the form $\mathbb{x} := (\mathtt{poly}, \mathbb{x}_{\mathtt{poly}})$ (resp. $\mathbb{x} := (\mathtt{opn}, \mathbb{x}_{\mathtt{opn}})$) is a $\mathtt{poly}$-instance (resp. $\mathtt{opn}$-instance).

Notice that if we have a CP-SNARK for $\mathcal{R}_{\mathtt{poly}}$ and a CP-SNARK for $\mathcal{R}_{\mathtt{opn}}$ we can easily define a single proof system that proves the relation $\hat{\mathcal{R}}$. In fact, the relation $\hat{\mathcal{R}}$ could be seen

as a join of the two relations plus some syntactic sugar. The reason to use a single CP-SNARK for $\hat{\mathcal{R}}$ (instead of one for each relation) is rather technical, and it has to do with the security guarantees when we compose protocols in the AGM [ABK+21]. Indeed, we need a CP-SNARK for $\mathcal{R}_{\mathtt{poly}}$ that is simulation-extractable in the AGM even in presence of the simulated proofs for a CP-SNARK for $\mathcal{R}_{\mathtt{opn}}$. In particular, the simulated proofs for the latter CP-SNARK could contain group elements that might interfere with the security proved for the former CP-SNARK (and vice versa). In other words, even if the first CP-SNARK is simulation-extractable (in the AGM), it could potentially be insecure when we use it in combination with the second CP-SNARK (if we are using the same group to instantiate the proof of opening).

Let now introduce the following set of policies $\hat{\mathbf{\Phi}}$.

Definition 3.6.6. *A policy $\hat{\Phi} := (\hat{\Phi}_0, \hat{\Phi}_1) \in \hat{\mathbf{\Phi}}$. $\Phi_0(\mathsf{pp}_{\mathbb{G}})$ outputs parameters pp_Φ that contain a set of vectors of polynomials $\mathcal{Q}_v = \{\vec{v}^{(i)} = (v_1^{(i)}, \ldots, v_n^{(i)})\}_{i \in \mathsf{poly}(\lambda)}$ and a list $\mathsf{coms} := (\vec{\mathsf{c}}^{(i)})_{i \in [q]}$ sampled from a distribution $\mathcal{D}$ where the $\mathcal{D}$-Aff-MDH assumption holds, while $\hat{\Phi}_1$ is defined as follows.*

- **Semi-adaptive w.r.t. `poly`-queries.** *Given the set of simulation queries $\mathcal{Q}_{\mathsf{sim}}$, define the projections of the set to the `poly`-simulation queries and `opn`-simulation queries. Let the i-th `poly`-query to the simulation oracle be $(\mathbb{x}, \mathsf{aux}, \pi) \in \mathcal{Q}_{\mathsf{sim}}$, and parse $\mathbb{x}$ as $(\mathtt{msg}, \vec{\mathsf{c}}, (G^{(k)}, \vec{v}^{(k)})_{k \in [n_e]})$.*

  1. *$\forall k \in [n_e] : \vec{v}^{(k)} \in \mathcal{Q}_v$*
  2. *parse aux as two matrices $\vec{C}, \vec{F}$; the tuple $(\vec{\mathsf{c}}, \vec{C}, \vec{F})$ satisfies the commitment check, namely $\vec{\mathsf{c}} = \vec{C} \cdot \vec{\mathsf{c}}^{(i)} + \vec{F} \cdot \mathsf{ck}$*
  3. *the view defined by set of all `poly`-queries satisfies the algebraic consistency for CP (cf. Definition 3.5.1).*

- **Extractor policy for `opn`-forgery.** *If the forgery of the adversary $(\mathbb{x}^*, \pi^*)$ is of the form $\mathbb{x}^* := (\mathtt{opn}, \vec{\mathsf{c}}^*)$, parse aux_Φ as (yet another) forgery $\tilde{\mathbb{x}} := (\mathtt{poly}, \mathtt{msg}, \vec{\mathsf{c}}, (G^{(k)}, \vec{v}^{(k)})_{i \in [n_e]}), \tilde{\pi}$ and check that:*

  1. *All the commitments $\vec{\mathsf{c}}^*$ are in the vector of commitments $\vec{\mathsf{c}}$,*
  2. *The proof $\tilde{\pi}$ is valid according to the extractor policy for `poly`-forgeries below.*

- **Extractor policy for `poly`-forgery.** *If the forgery of the adversary $(\mathbb{x}^*, \pi^*)$ is of the form $\mathbb{x}^* := (\mathtt{poly}, \mathbb{x}')$, then check that $(\mathbb{x}', \pi^*) \notin \mathcal{Q}_{\mathsf{sim}}$, i.e., π^* is "fresh" and not produced by the simulation oracle on input $\mathbb{x}'$. Moreover, there exists $k \in [n_e]$ such that for all j the polynomial $v_j^{(k)}$ has degree at least 1 (and degree $\mathsf{poly}(\lambda)$).*

Intuitively, the extractor policy for `opn`-forgery means that a proof of opening for a commitment c can be extracted (given some auxiliary information) only when the adversary can exhibit a proof of evaluation that involves such a commitment.

Definition 3.6.7 (Compilation-Ready CP-SNARK)**.** *A Compilation-Ready CP-SNARK is a $\hat{\mathbf{\Phi}}$-simulation-extractable CP-SNARK for $\hat{\mathcal{R}}$.*

$\underline{\Pi.\mathsf{Prove}(\mathsf{srs}, \mathsf{ek}_{\mathbb{i}}, \mathbb{x}, \mathbb{w})}$:

for $i \in [r(|\mathbb{i}|)]$ **do** :
  $(\vec{p}_i, \vec{\pi}_i) \leftarrow \mathsf{P}(\mathbb{F}, \mathbb{i}, \mathbb{x}, \mathbb{w}, \rho_1, \ldots, \rho_{i-1})$
  for $j \in [n(i)]$ **do** : $(\mathsf{c}_{i,j}, o_{i,j}) \leftarrow \mathsf{Com}(\mathsf{ck}, p_{i,j})$
  $\mathbb{x}_{o,i} \leftarrow (\mathtt{opn}, \vec{\mathsf{c}}_i)$, $\vec{\mathsf{c}}_i := (\mathsf{c}_{i,j})_j, \vec{o}_i := (o_{i,j})_j$
  $\pi_{\mathtt{opn},i} \leftarrow \mathsf{CP.Prove}(\mathsf{ck}, \mathbb{x}_{o,i}, (\vec{p}_i, \vec{o}_i)), \bar{\pi}_i := (\vec{\mathsf{c}}_i, \pi_{\mathtt{opn},i}, \vec{\pi}_i)$
  $\rho_i \leftarrow \mathsf{RO}(\mathsf{vk}_{\mathbb{i}}, \mathbb{x}, \bar{\pi}_1, \ldots, \bar{\pi}_i)$ // Fiat-Shamir transform
$(G^{(k)}, v^{(k)})_{k\in[n_e]} \leftarrow \mathcal{V}(\mathbb{F}, \mathbb{i}, \mathbb{x}, \rho_1, \ldots, \rho_{r-1})$
for $k \in [n_e]$ **do** : $\bar{G}^{(k)}(X, X_1, \ldots, X_n) \leftarrow G^{(k)}(X, X_1, \ldots, X_n, \vec{\pi})$
$\mathtt{trns} \leftarrow (\mathsf{vk}_{\mathbb{i}}, \mathbb{x}, \bar{\pi}_1, \ldots, \bar{\pi}_r)$
$\mathbb{x}_{\mathtt{poly}} \leftarrow (\mathtt{poly}, \mathtt{trns}, \vec{\mathsf{c}}, (\bar{G}^{(k)}, v_1^{(k)}, \ldots, v_n^{(k)})_{k\in[n_e]})$
$\pi_{\mathtt{poly}} \leftarrow \mathsf{CP.Prove}(\mathsf{srs}, \mathbb{x}_{\mathtt{poly}}, (\vec{p}, \vec{o}))$
return $(\vec{\mathsf{c}}, \vec{\pi}, (\pi_{\mathtt{opn},i})_{i\in[r]}, \pi_{\mathtt{poly}})$

$\underline{\Pi.\mathsf{Verify}(\mathsf{vk}_{\mathbb{i}}, \mathbb{x}, \pi_{\Pi})}$:

compute $(\bar{\pi}_1, \ldots, \bar{\pi}_r)$
for $i \in [r(|\mathbb{i}|) - 1]$ **do** : $\rho_i \leftarrow \mathsf{RO}(\mathsf{vk}_{\mathbb{i}}, \mathbb{x}, \bar{\pi}_1, \ldots, \bar{\pi}_{i-1})$
$(G^{(k)}, v^{(k)})_{k\in[n_e]} \leftarrow \mathcal{V}(\mathbb{F}, \mathbb{i}, \mathbb{x}, \rho_1, \ldots, \rho_{r-1})$
for $k \in [n_e]$ **do** : $\bar{G}^{(k)}(X, X_1, \ldots, X_n) \leftarrow G^{(k)}(X, X_1, \ldots, X_n, \vec{\pi})$
$\mathtt{trns} \leftarrow (\mathsf{vk}_{\mathbb{i}}, \mathbb{x}, \bar{\pi}_1, \ldots, \bar{\pi}_r)$
$\mathbb{x}_{\mathtt{poly}} \leftarrow (\mathtt{poly}, \mathtt{trns}, \vec{\mathsf{c}}, (\bar{G}^{(k)}, v_1^{(k)}, \ldots, v_n^{(k)})_{k\in[n_e]})$
return $\bigwedge_{i\in[r]} \mathsf{CP.Verify}(\mathsf{ck}, \mathbb{x}_{\mathtt{opn},i}, \pi_{\mathtt{opn},i}) \ \wedge \ \mathsf{CP.Verify}(\mathsf{srs}, \mathbb{x}_{\mathtt{poly}}, \pi_{\mathtt{poly}})$

$\underline{\Pi.\mathsf{Derive}(\mathsf{srs}, \mathbb{i})}$:

$\vec{p}_0 \leftarrow \mathsf{I}(\mathbb{F}, \mathbb{i})$;
for $j \in [n(|\mathbb{i}|, 0)]$ **do** :
  $\mathsf{c}_{0,j} \leftarrow \mathsf{Com}(\mathsf{ck}, p_{0,j}; \vec{0})$
return $(\vec{p}_0, (\mathsf{c}_{0,i})_{i\in[n(|\mathbb{i}|,0)]})$

Figure 3.3: The Compiler to Universal zkSNARKs.

In Section 3.6.3 we describe a simple and unoptimized CP-SNARK for $\hat{\mathcal{R}}$ for KZG, whose security analysis is given in the AGM. The CP-SNARK uses classical random-point evaluation for the polynomial equations and a *vacuous proof of opening* that can be extracted given the algebraic representation. The extraction of the latter is successful, namely the representation depends only on the element in the commitment key, only if the adversary exhibits a proof of polynomial equation. Both the proof of polynomial equation and of opening are extracted through the algebraic representations. Thus, unless the binding property is broken, when the proof of evaluation holds the proof of opening can be extracted correctly.

3.6.2 The Universal zkSNARK

Let Φ_{SE} be the standard (strong) simulation extractability policy. Namely, the policy checks that the forgery of the adversary is a tuple $(\mathbb{x}_{\Pi}, \pi_{\Pi}) \notin \mathcal{Q}_{\mathsf{sim}}$.

Theorem 3.6.1. *Let* CP *be a compilation-ready CP-SNARK (cf. Definition 3.6.7). Let* PIOP *be a PIOP for an indexed relation* $\mathcal{R}$ *that is state-restoration straight-line extractable (cf. Defini-*

tion 3.6.2), and bounded special honest-verifier zero-knowledge, where the technical condition of Definition 3.6.4 holds and equipped with a commitment simulator for CP *(cf. Definition 3.6.5). Let* Π *be the zkSNARK compiled from* PIOP *using the compiler from Fig. 3.3. Then* Π *is* $\Phi_{\mathtt{SE}}$*-simulation-extractable.*

We follow the classical compilation strategy where: for each of the r rounds, the zkSNARK prover sends commitments of the PIOP oracle polynomials (along with a proof of knowledge) and then computes the PIOP verifier's challenges using Fiat-Shamir; in the last round, the prover sends a CP proof that the PIOP verifier accepts, i.e., Eq. (2.2) holds w.r.t. all the commitments sent earlier. Notably, this CP proof is produced using the statement and the hash of the transcript as "message" for the signature of knowledge.

We briefly discuss how the properties of PIOP and CP play a role in the security of the compiled zkSNARK Π. We recall that in the simulation-extractability experiment, we have an adversary $\mathcal{A}$ who makes simulation queries for statements of its choice and eventually comes up with a forgery, which is a statement-proof that is new and valid. The goal is to show that for such adversary there is an extractor that outputs a valid witness with overwhelming probability. Roughly speaking, we build this extractor by first extracting the committed oracle polynomials from the CP "proof of knowledge" in the random oracle query of $\mathcal{A}$ in each round,[12] and then by running the PIOP extractor to obtain the witness.

For this extraction strategy to work, we need two conditions: (A) The "proof of knowledge" extraction must be valid. (B) The zkSNARK extractor feeds the PIOP extractor with polynomials that pass the PIOP verification equations. A technicality about relying on CP extraction for (A) and (B) is that we actually have to make a reduction to the its *policy-based simulation extractability*. In particular, this means that we have to turn $\mathcal{A}$ into CP adversaries that comply with the policy $\hat{\mathbf{\Phi}}$.

To obtain (A), we use the second property of $\hat{\mathbf{\Phi}}$ mentioned above, which ensures a valid extraction if the adversary later provides a valid proof of polynomial evaluation. This is however the case for us, since a successful adversary must provide such proof.

For (B), we rely on the following observations. If $\mathcal{A}$ produces a forgery for a new statement of Π then the CP proof (aka signature of knowledge) must use a new message, and thus we can build a CP adversary returning a new statement-proof pair. If $\mathcal{A}$ produces a forgery for a statement queried to the simulation oracle, then by strong simulation extractability the proof must be new, which means that: either the commitments in the transcript are different, or the commitments are all the same but the "PIOP verifier" proof is different. In the former case, we get a different transcript, which yields a CP forgery with a new message, as in the previous case. In the latter case, the transcript is the same, and we get a CP forgery with the same message but fresh proof. Notably, all the cases, the CP forgeries respect the degree-1 condition thanks to the compiler-safe property of the PIOP. Finally, the reduction CP adversaries that we build satisfy the first property of $\hat{\mathbf{\Phi}}$ thanks to the algebraic verifier property of PIOP, which allows us to precompute the instance-independent polynomials $\tilde{v}_j^{(k)}$, and to the programmability of the random oracle that allows us to pre-sample the verifier's challenges $\vec{\rho}$, define $v_j^{(k)}(X) = \tilde{v}_j^{(k)}(X, \vec{\rho})$, and later program the random oracle to use these coins $\vec{\rho}$.

Proof. We prove the theorem assuming the commitments are not hiding. The adaption to

[12] Note, this avoids rewinding, since extraction is performed in the same moment when the adversary sends the proof of knowledge through a RO call.

hiding commitments is straightforward. We start showing the zero-knowledge simulator for Π. The simulator is in Fig. 3.4. For the description of the simulator, we assume we can couple the leakage function $F_{\mathtt{poly}}$, which defines the $F_{\mathtt{poly}}$-leaky zero-knowledge of the CP-SNARK, with a function $\tilde{F}_{\mathtt{poly}}$ that upon input the instance outputs the set $\mathcal{L}_{\mathbb{x}}$ of evaluation points necessary to compute a simulated proof. The zero-knowledge guarantees of the simulator come from the zero-knowledge property of the PIOP, the zero-knowledge property of the CP and the simulator commitment property of $\mathcal{S}_{\mathsf{Com}}$ (see Definition 3.6.5). Notice that $\mathcal{S}_1$ might abort if $(\mathsf{vk}_{\mathbb{i}}, \mathbb{x}, \bar{\pi}_1, \ldots, \bar{\pi}_i)$ for some i was already queried to the random oracle. However, by simply assuming that the first message of P has $\omega(\log |\mathbb{F}|)$ min-entropy then the event that the simulator aborts happens with negligible probability.

$\underline{\mathcal{S}(0, \mathsf{pp}_{\mathbb{G}})}$:
$\mathsf{srs}, \mathsf{st}_{\mathsf{CP}} \leftarrow\!\$\ \mathsf{CP}.\mathcal{S}(0, \mathsf{pp}_{\mathbb{G}})$
$\mu \leftarrow 0$ // $\mathcal{S}_1$ queries counter
for $j \in [q]$ **do** :
  $\mathsf{coms}^{(j)} \leftarrow\!\$\ \mathcal{S}_{\mathsf{Com}}(0, \mathsf{ck})$
  $\vec{\rho}^{(j)} = (\rho_1^{(j)}, \ldots, \rho_{r-1}^{(j)}) \leftarrow\!\$\ \mathbb{F}^{r-1}$;
$\mathsf{st}_{\mathcal{S}} \leftarrow (\mathsf{st}_{\mathsf{CP}}, \mu, (\mathsf{coms}^{(j)}, \vec{\rho}^{(j)})_j)$
return $\mathsf{srs}, \mathsf{st}_{\mathcal{S}}$

$\underline{\mathcal{S}(2, \mathsf{st}_{\mathcal{S}}, s, \mathsf{aux})}$:
if $(s, \mathsf{aux}, a) \in \mathcal{Q}_{\mathsf{RO}}$:
  return $a, \mathsf{st}_{\mathcal{S}}$
$a \leftarrow\!\$\ \mathbb{F}$
$\mathcal{Q}_{\mathsf{RO}} \leftarrow \mathcal{Q}_{\mathsf{RO}} \cup (s, \mathsf{aux}, a)$
return $a, \mathsf{st}_{\mathcal{S}}$

$\underline{\mathcal{S}(1, \mathsf{st}_{\mathcal{S}}, \mathsf{srs}, (\mathbb{i}, \mathbb{x}))}$:
$\mathsf{st}_{\mathcal{S}} \leftarrow (\mathsf{st}_{\mathsf{CP}}, \mu, (\mathsf{coms}^{(j)}, \vec{\rho}^{(j)})_j)$
$\mathsf{coms} \leftarrow \mathsf{coms}^{(\mu)}$
$\vec{\pi}_1, \ldots, \vec{\pi}_r \leftarrow\!\$\ \mathsf{PIOP}.\mathcal{S}_0(\mathbb{F}, \mathbb{i}, \mathbb{x}, \vec{\rho}^{(\mu)}; r)$
$(G^{(k)}, v^{(k)})_{k \in [n_e]} \leftarrow \mathcal{V}(\mathbb{F}, \mathbb{x}, \vec{\rho}^{(\mu)})$
for $k \in [n_e]$ **do** : $\bar{G}^{(k)}(\vec{X}) \leftarrow G^{(k)}(\vec{X}, \vec{\pi})$
$\vec{M}_{\mathbb{i},\mathbb{x}} \leftarrow \mathcal{S}_{\mathsf{Com}}(1, \mathsf{ck}, |\mathbb{i}|, \mathbb{x})$
$\vec{\mathsf{c}} = \vec{M}_{\mathbb{i},\mathbb{x}} \cdot \mathsf{coms} \| \mathsf{ck}$
parse $\vec{\mathsf{c}} = \mathsf{vk}_{\mathbb{i}}, \vec{\mathsf{c}}_1, \ldots, \vec{\mathsf{c}}_r$
for $i \in [r]$ **do** :
  $\pi_{\mathtt{opn},i}, \mathsf{st}_{\mathsf{CP}} \leftarrow \mathsf{CP}.\mathcal{S}_1(\mathsf{st}_{\mathsf{CP}}, (\mathtt{opn}, \vec{\mathsf{c}}_i))$
  $\bar{\pi}_i = (\vec{\mathsf{c}}_i, \pi_{\mathtt{opn},i}, \vec{\pi}_i)$
  $\mathtt{trns} \leftarrow (\mathsf{vk}_{\mathbb{i}}, \mathbb{x}, \bar{\pi}_1, \ldots, \bar{\pi}_r)$
$\mathbb{x}_{\mathtt{poly}} \leftarrow (\mathtt{poly}, \mathtt{trns}, \vec{\mathsf{c}}, (G^{(k)}, v^{(k)})_{k \in [n_e]})$
$leak \leftarrow \mathsf{PIOP}.\mathcal{S}_1(\mathbb{F}, \mathbb{i}, \mathbb{x}, \tilde{F}_{\mathtt{poly}}(\mathbb{x}_{\mathtt{poly}}); r)$
$\pi_{\mathtt{poly}}, \mathsf{st}_{\mathsf{CP}} \leftarrow \mathsf{CP}.\mathcal{S}_1(\mathsf{st}_{\mathsf{CP}}, \mathbb{x}_{\mathtt{poly}}, leak)$
for $i \in [r-1]$ **do** :
  if $((\mathsf{vk}_{\mathbb{i}}, \mathbb{x}, \bar{\pi}_1, \ldots, \bar{\pi}_i), \cdot, \cdot) \in \mathcal{Q}_{\mathsf{RO}}$: **abort**
  $\mathcal{Q}_{\mathsf{RO}} \leftarrow \mathcal{Q}_{\mathsf{RO}} \cup ((\mathbb{x}, \bar{\pi}_1, \ldots, \bar{\pi}_i), \cdot, \rho_i)$
$\mathsf{st}_{\mathcal{S}} \leftarrow (\mathsf{st}_{\mathsf{CP}}, \mu + 1, (\mathsf{coms}^{(j)}, \vec{\rho}^{(j)})_j)$
$\pi \leftarrow (\bar{\pi}_1, \ldots, \bar{\pi}_r, \pi_{\mathtt{poly}})$
return $\pi, \mathsf{st}_{\mathcal{S}}$

Figure 3.4: The simulator $\mathcal{S}$ for Π.

We define the extractor for Π for a given adversary $\mathcal{A}_{\Pi}$.

We make some simplifying assumptions on the behavior of $\mathcal{A}_{\Pi}$: (1) the adversary always queries first the RO on a string that can be parsed as $(\mathbb{i}, \mathbb{x})$ before querying the simulation oracle on the same string, (2) the auxiliary string $\mathsf{aux}_{\mathcal{E}}$ output by $\mathcal{A}_{\Pi}$ can be parsed as a list of strings $(s_i, \mathsf{aux}_i, \mathsf{st}_i)_i$ and a string $\mathsf{aux}'_{\mathcal{E}}$ where for any i we have $(s_i, \mathsf{aux}_i, \mathsf{st}_i)$ string is identical

to the auxiliary input output at the i-th query of the adversary and (3) for any i the auxiliary input of the i-th random oracle query of the adversary contains the full internal state of the adversary. These assumptions are w.l.g. In fact, given an adversary $\mathcal{A}_\Pi$ that does not respect these rules we can always define another adversary that runs internally $\mathcal{A}_\Pi$, collects all the necessary information to comply with (2) and (3), and moreover follows the rule (1).

Let $\mathcal{B}_i$ be a reduction to the simulation extractability of CP that runs $\mathcal{A}_\Pi$ (simulating the oracles for $\mathcal{A}_\Pi$ using its own oracles and the code defined in Fig. 3.4) and sets its output to be the i-th random oracle query of $\mathcal{A}_\Pi$. For any $i \in [q]$ let $\mathcal{E}_i$ be the extractor associated to $\mathcal{B}_i$ (thanks to the $\hat{\Phi}$-simulation extractability of CP we can associate to $\mathcal{B}_i$ an extractor, in Lemma 3.6.2 we describe $\mathcal{B}_i$ with more details, and we show that it complies with the $\hat{\Phi}$ policy). Similarly, let $\mathcal{E}_{\mathtt{poly}}$ be the extractor for the adversary $\mathcal{B}_{\mathtt{poly}}$ that runs $\mathcal{A}_\Pi$ (simulating the oracles of $\mathcal{A}_\Pi$ using its own oracles and the code defined in Fig. 3.4) until completion, and isolates the instance $\mathbb{x}_{\mathtt{poly}} \coloneqq (\mathtt{poly}, \mathtt{trns}, \vec{\mathsf{c}}, (G^{(k)}, \vec{v}^{(k)})_{k\in[n_e]})$ and the proof $\pi_{\mathtt{poly}}$ in the forgery of $\mathcal{A}_\Pi$, and outputs all the auxiliary outputs that $\mathcal{A}_\Pi$ does.

We first set some notation:

- Let $\mathcal{P}_i$ be the indexes of the polynomials sent at the i-th round by the PIOP.
- Given a proof π for Π we define *the RO-queries set of* π be the set of string $\{(\mathsf{vk}_{\mathbb{i}}, \mathbb{x}), \ldots, (\mathsf{vk}_{\mathbb{i}}, \mathbb{x}, \bar{\pi}_1, \ldots, \bar{\pi}_r)\}$.
- We say that a proof π *shares a simulated commitment* c of proof π' simulated by $\mathcal{S}$ if in $\mathsf{st}_{\mathcal{S}}$ one can find $\vec{m}_1 \neq \vec{0}$ and $\vec{m}_2$ such that $\mathsf{c} = \vec{m}_1 \| \vec{m}_2 \cdot \mathsf{coms} \| \mathsf{ck}$.

The extractor $\mathcal{E}_\Pi(\mathbb{x}_\Pi, \pi_\Pi, \mathsf{aux}_{\mathcal{E}})$:

1. Parse $\mathsf{aux}_{\mathcal{E}}$ as the concatenation of a list $(s_i, \mathsf{aux}_i, \mathsf{st}_i)_i$ and $\mathsf{aux}'_{\mathcal{E}}$, where $(s_i, \mathsf{aux}_i, \mathsf{st}_i)$ is the output of $\mathcal{A}_\Pi$ at the i-th query to the ROM and $\mathsf{aux}'_{\mathcal{E}}$ the remaining auxiliary information given by the adversary (namely, the auxiliary information associated with its last output).
2. From π_Π derive the messages $\bar{\pi}_1, \ldots, \bar{\pi}_r$ and find the indexes q_i such that $s_{q_i} = (\mathsf{vk}_{\mathbb{i}}, \mathbb{x}, \bar{\pi}_1, \ldots, \bar{\pi}_i)$.
3. If π_Π shares a simulated commitment with one of the simulated proofs then return $\perp$.
4. For $i \in [r]$ run $\mathbb{w}_{i,\mathtt{opn}} \leftarrow \mathcal{E}_{q_i}(\mathsf{srs}, \vec{\mathsf{c}}_i, \pi_{\mathtt{opn},i}, \mathsf{aux}_i)$ where $\bar{\pi}_i$ contains both the instance and the proof of opening and $\mathbb{w}_{i,\mathtt{opn}} = (p'_j)_{j\in\mathcal{P}_i}$.
5. Run $(p_j)_{j\in[n]} \leftarrow \mathcal{E}_{\mathtt{poly}}(\mathsf{srs}, \mathbb{x}_{\mathtt{poly}}, \pi_{\mathtt{poly}}, \mathsf{aux}')$.
6. If $\exists i : p'_i \neq p_i$ then return $\perp$.
7. If $\exists i \in [n]$ and $j \in \mathcal{P}_i : \mathsf{VerCom}(\mathsf{ck}, \mathsf{c}_j, p'_j) \neq 1$ then return $\perp$.
8. If $\exists k : G^{(k)}(X, p_1(v_1^{(k)}(X)), \ldots, p_n(v_n^{(k)}(X)), \pi_1, \ldots, \pi_r) \not\equiv 0$ then return $\perp$.
9. Return $\mathcal{E}_{\mathsf{PIOP}}(\mathbb{i}, \mathbb{x}_\Pi, (p_j)_{j\in[n]})$.

To analyze the success of the extractor we define a series of hybrid games. We start from the first hybrid that is the $\mathbf{Exp}^{sr}_{\mathcal{A}_{\mathsf{PIOP}},\mathsf{PIOP}}(\mathbb{F})$ experiment for PIOP (see Definition 3.6.2) for an adversary $\mathcal{A}_{\mathsf{PIOP}}$ that we define next.

underline$\mathcal{A}_{\mathsf{PIOP}}$:

1. Run simulator $\mathsf{srs}, \mathsf{st}_{\mathcal{S}} \leftarrow \mathcal{S}(0, \mathsf{pp}_{\mathbb{G}})$ and set $\mathcal{Q}_{\mathsf{RO}}, \mathcal{Q}_{\mathsf{sim}}$ empty sets.
2. Run $\mathcal{A}_{\Pi}(\mathsf{srs})$ and answer all the simulation queries of $\mathcal{A}_{\Pi}$ with $\mathcal{S}_1$.
3. Upon i-th query (s_i, aux_i) from $\mathcal{A}_{\Pi}$ to $\mathcal{S}_2$:
   (a) if s_i is in the RO-queries set of a simulated proof in $\mathcal{Q}_{\mathsf{sim}}$ then run $\mathcal{S}_2$ on input s_i.
   (b) Else parse s_i as a (partial) transcript $\mathtt{trns} = (\mathsf{vk}_{\mathbb{i}}, \mathbb{x}, \bar{\pi}_1, \ldots, \bar{\pi}_{r'})$ for $r' \in [r]$; parse $\bar{\pi}_j$ as $(\vec{\mathsf{c}}_j, \pi_{\mathsf{opn},j}, \vec{\pi}_j)$; compute and parse as polynomials $\mathbb{w} \leftarrow \mathcal{E}_i(\mathsf{srs}, \vec{\mathsf{c}}_{r'}, \pi_{\mathsf{opn},r'}, \mathsf{aux}_i)$; find in $\mathsf{SeenStates}$ the state $\mathsf{cvs} = (\mathbb{i}, \mathbb{x}, \vec{\pi}_1, \{p_{1,i}\}_i \| \rho_1 \| \ldots \| \vec{\pi}_{r'-1}, \{p_{r'-1,i}\}_i \| \rho_{r'-1})$, set the verifier state to cvs, and send the message $(\mathbb{w}, \vec{\pi}_{r'})$ to the PIOP verifier. Receive the challenge $\rho_{r'}$, and program the random oracle adding $(s_i, \mathsf{aux}_i, \rho_{r'})$ to $\mathcal{Q}_{\mathsf{RO}}$.
4. Eventually the adversary outputs a (valid) forgery $(\mathbb{x}_{\Pi}, \pi_{\Pi})$. From π_{Π} derive the PIOP transcript $\mathtt{trns} \coloneqq (\mathbb{i}, \mathbb{x}, \bar{\pi}_1, \rho_1, \ldots, \bar{\pi}_r)$, and act as if $\mathcal{A}$ has queried $\mathcal{S}_1$ with $(\mathtt{trns}, \cdot)$: i.e., let i be the index of RO query of the partial transcript $(\mathbb{i}, \mathbb{x}, \bar{\pi}_1, \rho_1, \ldots, \bar{\pi}_{r-1})$; as described in the previous step, find the cvs in $\mathsf{SeenStates}$ associated with s_i, set the verifier state to cvs, extract the (last) witness $\mathbb{w}$ and send $(\mathbb{w}, \vec{\pi}_r)$ to the verifier. cvs and the last messages $(\mathbb{w}, \vec{\pi}_r)$ define a full transcript: this would trigger the verifier to perform the decision phase of the PIOP and set the decision bit d of the game.

Let $\mathbf{H}_0$ be the $\mathbf{Exp}^{sr}_{\mathcal{A}_{\mathsf{PIOP}},\mathsf{PIOP}}(\mathbb{F})$. By the state-restoration knowledge extractability of PIOP:

$$\Pr[\mathbf{H}_0] \in \mathsf{negl}(|\mathbb{F}|).$$

Consider $\mathbf{H}_1$ that additionally extracts $(p_j)_{j\in[n]} \leftarrow \mathcal{E}_{\mathtt{poly}}(\mathsf{srs}, \mathbb{x}_{\mathtt{poly}}, \pi_{\mathtt{poly}}, \mathsf{aux}')$ and returns 1 if $\exists k : G^{(k)}(X, p_1(v_1^{(k)}(X)), \ldots, p_n(v_n^{(k)}(X)), \pi_1, \ldots, \pi_r) \not\equiv 0$ or $\exists j : \mathsf{VerCom}(\mathsf{ck}, \mathsf{c}_j, p_j) = 0$.

Lemma 3.6.1. $\Pr[\boldsymbol{H}_1] \leq \Pr[\boldsymbol{H}_0] + \epsilon_{\mathsf{CP}}$

Proof. We reduce to $\hat{\Phi}$-simulation extractability of CP. We define with more details the adversary $\mathcal{B}_{\mathtt{poly}}$ that runs $\mathcal{A}_{\Pi}$, and we define a policy $\hat{\Phi}_0$ that samples parameters for $\mathcal{B}_{\mathtt{poly}}$.

Policy $\hat{\Phi}_0(\mathsf{pp}_{\mathbb{G}})$

1. Run $\mathcal{S}(0, \mathsf{pp}_{\mathbb{G}})$
2. Parse $\mathsf{st}_{\mathcal{S}} = (\mathsf{st}_{\mathsf{CP}}, 0, (\mathsf{coms}^{(j)})_j, (\vec{\rho}^{(j)})_j)$.
3. Let $\mathcal{Q}_v$ be an empty set. Let D be the maximum degree of the polynomials in srs. For $d \in [D]$:
   (a) let $(\tilde{v}_j^{(k)})_{j,k} \leftarrow \tilde{\mathcal{V}}(\mathbb{F}, d)$ be the polynomials defined by the (non-adaptive) algebraic verifier of PIOP (see Definition 3.6.1)
   (b) add to $\mathcal{Q}_v$ the polynomials $\tilde{v}_j^{(k)}(X, \vec{\rho}') : i \in [q], j \in [n], k \in [n_e]$

Next, we define the reduction $\mathcal{B}_{\mathtt{poly}}$.

Reduction $\mathcal{B}_{\mathtt{poly}}(\mathsf{srs}, \mathsf{pp}_{\hat{\Phi}})$

1. Run $\mathcal{A}_{\Pi}(\mathsf{srs})$.
2. Upon query $((\mathbb{i}, \mathbb{x}), \mathsf{aux})$ to the simulation oracle, run the same strategy of $\mathcal{S}_1$ defined in Fig. 3.4 where the calls to $\mathsf{CP}.\mathcal{S}_1$ are forwarded to the simulation oracle of $\mathcal{B}_{\mathtt{poly}}$.
3. Given the forgery $((\mathbb{i}, \mathbb{x}_{\Pi}), \pi_{\Pi})$ output by $\mathcal{A}_{\Pi}$, define the instance $\mathbb{x}_{\mathtt{poly}} = (\mathtt{poly}, \mathtt{trns}, \vec{\mathsf{c}}, (G^{(k)}, \vec{v}^{(k)})_{k\in[n_e]})$ and the proof $\pi_{\mathtt{poly}}$.
4. Return the forgery $(\mathbb{x}_{\mathtt{poly}}, \pi_{\mathtt{poly}})$ and set the auxiliary input $\mathsf{aux}_{\mathcal{E}}$ as the adversary $\mathcal{A}_{\Pi}$ does.

By inspection, if the forgery of $\mathcal{A}_{\Pi}$ passes the verification then forgery of $\mathcal{B}_{\mathtt{poly}}$ passes the verification too. Moreover, by the (strong) simulation extractability game of $\mathcal{A}_{\Pi}$ we have that $((\mathbb{i}, \mathbb{x}_{\Pi}), \pi_{\Pi})$ is not in the set of simulation proofs $\mathcal{Q}_{\mathsf{sim},\Pi}$ of $\mathcal{A}_{\Pi}$, thus the pair $(\mathbb{x}_{\mathtt{poly}}, \pi_{\mathtt{poly}})$ is not in the simulation proofs $\mathcal{Q}_{\mathsf{sim},\mathtt{poly}}$ of $\mathcal{B}_{\mathtt{poly}}$. In particular, there are three cases:

1. If $\mathcal{A}$ never queried $(\mathbb{i}, \mathbb{x}_{\Pi})$ to its simulation oracle, i.e. $\mathbb{x}_{\Pi}$ is a "fresh" instance for Π, then $\mathcal{B}_{\mathtt{poly}}$ never queried the simulator with $\mathbb{x}_{\mathtt{poly}}$ (notice that $\mathbb{x}_{\mathtt{poly}}$ contains $\mathbb{x}_{\Pi}$ in the message $\mathtt{trns}$).

2. Otherwise, $\mathcal{A}$ queried $(\mathbb{i}, \mathbb{x}_{\Pi})$ to its simulation oracle:

   (a) For any simulated proof π'_{Π} for $(\mathbb{i}, \mathbb{x}_{\Pi})$ the transcript of π_{Π} is different from the simulated one. This implies that $\mathbb{x}_{\mathtt{poly}} \neq \mathbb{x}'_{\mathtt{poly}}$ (for the same reason of above).

   (b) There exists a simulated proof π'_{Π} for $(\mathbb{i}, \mathbb{x}_{\Pi})$ such that the transcripts of the forgery and of the simulated proof are equal. Since the forgery of $\mathcal{A}_{\Pi}$ is not in $\mathcal{Q}_{\mathsf{sim},\Pi}$ then the proofs $\pi_{\mathtt{poly}} \neq \pi'_{\mathtt{poly}}$.

By the property of the PIOP we have that there exists an index k such that for all j we have $v_j^{(k)}$ is not constant polynomial, thus the forgery of $\mathcal{B}_{\mathtt{poly}}$ meets the extractor policy.

Finally, by the definition of the simulator $\mathcal{S}$ in Fig. 3.4, the query to the simulation oracle of $\mathcal{B}_{\mathtt{poly}}$ respects the simulator policy of $\hat{\Phi}_1$. In fact, the simulator policy of $\hat{\Phi}$ matches the constraint in Definition 3.6.5. Thus, the reduction $\mathcal{B}_{\mathtt{poly}}$ follows the policy $\hat{\Phi}$. On the other hand, the distinguish event implies that the extractor fails, thus this would break the $\hat{\Phi}$-simulation extractability of CP. □

Let $\mathbf{H}_2$ additionally return 1 if $\exists i, j : j \in \mathcal{P}_i \wedge \mathsf{VerCom}(\mathsf{ck}, \mathsf{c}_j, p'_j) \neq 1$, where q_i is the index of the random oracle query such that $(s_{q_i}, \mathsf{aux}_{q_i}, \rho_i) \in \mathcal{Q}_{\mathsf{RO}}$, and $\mathbb{w}_{q_i} = (p'_j)_{j\in\mathcal{P}_i}$ are the polynomials extracted by $\mathcal{E}_{q_i}$.

Lemma 3.6.2. $\Pr[\boldsymbol{H}_2] \leq \Pr[\boldsymbol{H}_1] + r \cdot \epsilon_{\mathsf{CP}}$

Proof. We prove through a series of r hybrids. Let $\mathbf{H}_{1,0}$ be the same as $\mathbf{H}_1$; moreover, let $\mathbf{H}_{1,i}$ be the same as $\mathbf{H}_{1,i-1}$, but that additionally returns 1 if $\exists j \in \mathcal{P}_i : \mathsf{VerCom}(\mathsf{ck}, \mathsf{c}_j, p'_j) \neq 1$, where q_i is the index of the random oracle query such that $(s_{q_i}, \mathsf{aux}_{q_i}, \rho_i) \in \mathcal{Q}_{\mathsf{RO}}$, and $\mathbb{w}_{q_i} = (p'_j)_{j\in\mathcal{P}_i}$ are the polynomials extracted by $\mathcal{E}_{q_i}$. Clearly, $\mathbf{H}_{1,r} \equiv \mathbf{H}_2$ (where r is the number of rounds of the PIOP).

We reduce again to $\hat{\Phi}$-simulation extractability of CP. We define the adversary $\mathcal{B}$ that runs $\mathcal{A}_\Pi$. The adversary $\mathcal{B}$ corresponds to the reduction $\mathcal{B}_{q_i}$ mentioned before. Additionally, we need to define a policy $\hat{\Phi}_0$ that samples the parameters for $\mathcal{B}$, we set the policy identical to the one used in Lemma 3.6.1.

Reduction $\mathcal{B}(\mathsf{srs}, \mathsf{pp}_{\hat{\Phi}})$

1. Run $\mathcal{A}_\Pi(\mathsf{srs})$.
2. Upon query $((\mathbb{i}, \mathbb{x}), \mathsf{aux})$ to simulation oracle, run the same strategy of $\mathcal{S}_1$ defined in Fig. 3.4 with the only difference that instead of calling $\mathsf{CP}.\mathcal{S}_1$ it makes a query to its simulator.
3. Parse the q_i-th query $(s_{q_i}, \mathsf{aux}_{q_i})$ to the RO of $\mathcal{A}_\Pi$ as a partial transcript where $\bar{\pi}_i := (\vec{\mathsf{c}}_i, \pi_{\mathtt{opn},i}, \vec{\pi}_i)$.
4. Continue running the adversary, parse the forgery output by $\mathcal{A}_\Pi$ as $(\bar{\pi}'_1, \ldots, \bar{\pi}'_r, \pi_{\mathtt{poly}})$. Set $\tilde{\mathbb{x}} := (\mathtt{poly}, \mathtt{trns}, \vec{\mathsf{c}}, (G^{(k)}, \vec{v}^{(k)})_{k\in[n_e]})$ as the verifier would do.
5. Return the forgery $((\mathtt{opn}, \vec{\mathsf{c}}_i), \pi_{\mathtt{opn},i})$, using as auxiliary information $\mathsf{aux}_{\mathcal{E}} \leftarrow \mathsf{aux}_{q_i}$ and $\mathsf{aux}_\Phi \leftarrow (\tilde{\mathbb{x}}, \pi_{\mathtt{poly}})$.

By inspection, if the forgery of $\mathcal{A}_\Pi$ passes the verification (both for $\mathtt{poly}$-instance in the final proof and for the $\mathtt{opn}$-instance in the q_i-th query) then $\mathcal{B}$'s forgery passes the verification too. Moreover, by the (strong) simulation extractability game of $\mathcal{A}_\Pi$ we have that $((\mathbb{i}, \mathbb{x}_\Pi), \pi_\Pi)$ is not in the set of simulations $\mathcal{Q}_{\mathsf{sim}}$ of $\mathcal{A}_\Pi$, thus the pair $(\tilde{\mathbb{x}}, \pi_{\mathtt{poly}})$ is not in the simulations of $\mathcal{B}$ (cf. Lemma 3.6.1). By the property of the PIOP, there exists an index k such that for all j we have $v_j^{(k)}$ is not constant polynomial, thus the forgery of $\mathcal{B}$ meets the extractor policy.

Finally, by the definition of the simulator $\mathcal{S}$ in Fig. 3.4, the query to the simulation oracle of $\mathcal{B}$ respects the simulator policy of $\hat{\Phi}_1$. In fact, the simulator policy of $\hat{\Phi}$ matches with the constraint in Definition 3.6.5. Notice we are running the experiment for $\mathcal{B}$ with the same extractor of $\mathcal{B}_{q_i}$. However, the first three outputs of $\mathcal{A}$ and $\mathcal{B}$ are distributed equivalently ($\mathcal{B}$ additionally outputs aux_Φ), thus the same extractor $\mathcal{E}_{q_i}$ works either for $\mathcal{B}_{q_i}$ or for $\mathcal{B}$. □

Let $\mathbf{H}_3$ additionally return 1 if $\exists i : p'_i \neq p_i$ where $(p'_j)_{j\in\mathcal{P}_i}$ are the polynomials extracted by $\mathcal{E}_{q_i}$ and $(p_j)_{j\in[n]}$ are the polynomials extracted by $\mathcal{E}_{\mathtt{poly}}$.

Lemma 3.6.3. $\Pr[\boldsymbol{H}_3] \leq \Pr[\boldsymbol{H}_2] + \epsilon_{\mathsf{CP}}$.

Proof. The distinguishing event implies that we can define an adversary $\mathcal{B}$ that runs the $\hat{\Phi}$-simulation extractability experiment for CP and finds a commitment c and two distinct polynomials p and p' such that $\mathsf{VerCom}(\mathsf{ck}, \mathsf{c}, p) = \mathsf{VerCom}(\mathsf{ck}, \mathsf{c}, p') = 1$[13].Let D be the maximum degree allowed by the commitment key, let $x \in \mathbb{F}$ such that $p(x) \neq p'(x)$ and consider the following (false) statement for $\mathcal{R}_{\mathtt{poly}}$ (and in turn of $\hat{\mathcal{R}}$):

- For $j \in [D+1]$ set $G^{(j)}(X, X_1, X_j) = (X_1 - p(j)) - (X - j)X_j$. Namely, the polynomial associated with formal variable X_1 evaluates on $p(j)$ at point j, and X_j is the *witness*.

[13]Intuitively, this breaks the binding property of the commitment scheme, however, the binding property does not assume that the adversary can see simulated proofs thus we cannot reduce to it. Here, instead, we show how forge a proof for an instance that is not in the language.

- Set $G^{(D+2)}(X, X_1, X_{D+2}) = (X_1 - p'(x)) - (X - x)X_{D+2}$. Namely, the polynomial associated to formal variable X_1 evaluates on $p'(x)$ at point x, and X_{D+2} is its *witness*.

We can create a valid proof for such a statement using the prover of CP: the first $d+1$ equations are proved using p and the polynomials $p_j(X) = (p(X) - p(j))/(X - j)$, while to prove the last equation we use the witness p'. Here, we use the hypothesis that CP uses internally a CP-SNARK for $\mathcal{R}_{\mathtt{m\text{-}evl}}$. The proof is for a statement that is not in the language, thus we break the simulation extractability of CP. □

Let $\mathbf{H}_4$ additionally return 1 if the forged proof shares a simulated commitment with a simulated proof in $\mathcal{Q}_{\mathsf{sim}}$.

Lemma 3.6.4. $\Pr[\boldsymbol{H}_4] \leq \Pr[\boldsymbol{H}_3] + \epsilon_{\mathsf{owf}}$

Proof. We need to bound the probability that $\mathcal{A}_{\Pi}$ returns a forgery that shares a simulated commitment with a proof in $\mathcal{Q}_{\mathsf{sim}}$.

Let $\mathcal{B}$ be the same reduction described in Lemma 3.6.1: as shown before, if $\mathcal{A}_{\Pi}$ satisfies the simulation-extractability policy then the reduction $\mathcal{B}$ satisfies the policy $\hat{\Phi}$. We notice that the extractor of $\mathcal{B}$ extracts a witness for all the commitments in $\vec{\mathsf{c}}$ in the forgery. Let assume that the forgery shares a commitment with the i-th simulated proof for some $i \in [q]$ and $\mathsf{coms} \leftarrow \vec{M}_{\mathbb{i},\mathbb{x}} \cdot (\mathsf{coms}^{(i)} \| \mathsf{ck})$ (in case the commitments are not homomorphic we assume that $\vec{M}_{\mathbb{i},\mathbb{x}}$ is the identity matrix). Since $\mathsf{coms} \leftarrow \vec{M}_{\mathbb{i},\mathbb{x}} \cdot (\mathsf{coms}^{(i)} \| \mathsf{ck})$ there exists a row $\vec{m}$ of the matrix $\vec{M}_{\mathbb{i},\mathbb{x}}$ such that $\mathsf{c}_j = \vec{m} \cdot \mathsf{coms}^{(i)}$. Moreover, we have a valid opening for c_j so $\mathsf{VerCom}(\mathsf{ck}, \mathsf{c}_j, p)$. Notice that a commitment function defines a one-way function; moreover, the simulated commitment is sampled from a distribution that is computationally indistinguishable from a distribution that samples *random* commitments. Thus, such an extractor would break the one-way property of the commitment function. □

Lemma 3.6.5. $\Pr[\boldsymbol{H}_4] = \mathbf{Adv}^{\Phi_{\mathtt{SE}}\text{-}se}_{\mathcal{A}_{\Pi},\mathcal{S},\mathcal{E}_{\Pi}}(\lambda)$

Proof. We now show that the probability that $\mathbf{H}_4$ outputs 1 is equal to the probability that the adversary $\mathcal{A}_{\Pi}$ wins the $\Phi_{\mathtt{SE}}$-se experiment against $\mathcal{E}_{\Pi}$. First, notice that the srs is generated by $\mathcal{S}(0, \mathsf{pp}_{\mathbb{G}})$ in both experiments. Upon (valid) forgery $((\mathbb{i}, \mathbb{x}_{\Pi}), \pi_{\Pi})$, we notice that:

- π_{Π} cannot share a simulated commitment with one of the simulated proofs (see Item 3) because of the check introduced in Lemma 3.6.4. Thus, all the RO queries of $\mathcal{A}_{\Pi}$ that constitute π_{Π} (namely the queries $q_1, \ldots, q_r$) were forwarded to the challenger in Item 3b (in particular, any of the query was already answerd by the programming of the RO by the simulator $\mathcal{S}_1$). This implies that the complete transcript sent by $\mathcal{A}_{\mathsf{PIOP}}$ is in the list $\mathsf{SeenStates}$.

  On the other hand, the extractor $\mathcal{E}_{\Pi}$ does not abort at Item 3.

- $\forall j \in [n]$, the polynomial p_j extracted by $\mathcal{E}_{q_i}$ is equal to p'_j extracted from $\mathcal{E}_{\mathtt{poly}}$ and $\mathsf{VerCom}(\mathsf{ck}, \mathsf{c}_j, p_j) = 1$; this is ensured by the checks introduced in Lemma 3.6.2 and Lemma 3.6.3. This implies that the extractor $\mathcal{E}_{\mathsf{PIOP}}$ in the $\mathbf{Exp}^{sr}_{\mathcal{A}_{\mathsf{PIOP}},\mathsf{PIOP}}$ in $\mathbf{H}_3$ is fed with the same polynomials extracted by $\mathcal{E}_{\Pi}$.

- Finally, the polynomial check on $\mathbb{x}_{\mathtt{poly}}$ must be satisfied by the extracted polynomials p_j because of the check introduced in Lemma 3.6.1. Thus, if the proof π_{PIOP} is valid, the decision bit in the state-restoration game is 1.

□

Having bound the probability that the adversary $\mathcal{A}_\Pi$ wins the $\Phi_{\mathtt{SE}}$-se experiment against $\mathcal{E}_\Pi$ concludes the proof. □

3.6.3 The Compilation-Ready CP-SNARK in the AGM

To connect together Section 3.5 and the results of this section, we show a simple compilation-ready CP-SNARK in the ROM based on batched KZG evaluation proofs (cf. Section 3.6.3).

Upon $\mathtt{opn}$-instances $\mathbb{x} := (\mathtt{opn}, \mathbb{x})$, the prover simply outputs an empty string (the verifier is the obvious algorithm).

Upon $\mathtt{poly}$-instances $\mathbb{x} := (\mathtt{poly}, \mathbb{x})$, where $\mathbb{x}' := (\mathtt{msg}, \vec{\mathsf{c}}, (\bar{G}^{(k)}, v_1^{(k)}, \ldots, v_n^{(k)})_{k\in[n_e]})$:

- $\mathsf{Prove}(\mathsf{ck}, \mathbb{x} = (\mathtt{poly}, \mathbb{x}'), \mathbb{w} = (\vec{p}, \vec{o}))$:
  1. Compute $x^* \leftarrow \mathsf{RO}(\mathbb{x}')$.
  2. Compute for any j, k the value $x_j^{(k)} \leftarrow v_j^{(k)}(x^*)$.
  3. Let $\mathcal{X} := \{x_1^*, \ldots, x_m^*\}$ be the set of the points $x_j^{(k)}$ for all k and j computed at the previous step, and let $\mathcal{P}_i := \{j : v_j^{(k)}(x^*) = x_i^*\}$.
  4. For $i \in [m]$, compute $\pi_{\mathtt{m\text{-}evl},i} \leftarrow \mathsf{P}_{\mathtt{m\text{-}evl}}(\mathsf{ck}, \mathbb{x}_{\mathtt{m\text{-}evl}}, (p_j, o_j)_{j\in\mathcal{P}_i})$ for the instance $\mathbb{x}_{\mathtt{m\text{-}evl}} := (x_i^*, (\mathsf{c}_j, p_j(x_i^*))_{j\in\mathcal{P}_i})$.
  5. Output $\left(\pi_{\mathtt{m\text{-}evl},i}, (p_j(x_i^*))_{j\in\mathcal{P}_i}\right)_{i\in[m]}$.
- $\mathsf{Verify}(\mathsf{ck}, \mathbb{x} = (\mathtt{poly}, \mathbb{x}'), \pi)$
  1. Compute $x^* \leftarrow \mathsf{RO}(\mathbb{x}')$.
  2. Compute for any j, k the value $x_j^{(k)} \leftarrow v_j^{(k)}(x^*)$.
  3. Let $\mathcal{X} := \{x_1^*, \ldots, x_m^*\}$ be the set of the points $x_j^{(k)}$ computed at the previous step, and let $\mathcal{P}_i := \{j : v_j^{(k)}(x^*) = x_i^*\}$.
  4. Parse π as $\left(\pi_{\mathtt{m\text{-}evl},i}, (y_{i,j})_{j\in\mathcal{P}_i}\right)_{i\in[m]}$. For any j, k let $\bar{y}_j^{(k)}$ be the (claimed) evaluation of the polynomial committed in c_j on point $v_j^{(k)}(x^*) \in \mathcal{X}$, this value is equivalent to $y_{i,j}$ for the index i such that $v_j^{(k)}(x^*) = x_i^*$.
  5. Output 1 if and only if:
     (a) $\forall i \in [m] : \mathsf{V}_{\mathtt{m\text{-}evl}}(\mathsf{ck}, \mathbb{x}_{\mathtt{m\text{-}evl},i} = (x_i^*, (\mathsf{c}_j, y_{i,j})_{j\in\mathcal{P}_i}), \pi_{\mathtt{m\text{-}evl},i}) = 1$
     (b) $\forall k \in [n_e] : \bar{G}^{(k)}(x^*, \bar{y}_1^{(k)}, \ldots, \bar{y}_n^{(k)}) = 0$.

For a "PIOP verifier" statement, the prover RO-hashes the instance and obtains a random point ξ, evaluates the polynomials $v_j^{(k)}(\xi)$ for any j and outputs the evaluations $p_j(v_j^{(k)}(\xi))$ together with a batch evaluation proof for all of them. For a "proof of knowledge" statement, the prover does not output an explicit proof element (we call this a *vacuous proof*), and we rely on the AGM to argue its extractability. The idea is that, for an algebraic adversary that produces an alleged commitment c and its algebraic representation, we can find a way to *open* c, under some

circumstances. For example, consider the adversary that, during the simulation-extractability experiment, hashes (i.e., makes a random oracle query) the commitment c, and later includes c in a "PIOP verifier" instance. Then the algebraic representation of c returned at hashing time must coincide with the same polynomial extracted at forgery time, otherwise one can break the standard binding of the commitment. Crucially, this scenario fits exactly the second part of the policy $\hat{\mathbf{\Phi}}$.

As for the third part of the policy, we notice that an attack similar to the mix-and-match malleability attack mentioned in the introduction applies for our compilation-ready CP-SNARK. For example, the adversary could ask a simulation for an instance that tests two (fake) commitments on constant values defined by the $v_j^{(k)}$, and then it can produce a forgery which includes one of the commitments by copying part of the simulated proof. Intuitively, this is why we require that the $v_j^{(k)}$ have degree at least 1: when evaluated on a fresh random point ξ, a valid proof for $p_j(v_j^{(k)}(\xi))$ intuitively ensures that the prover knows p_j

Definition 3.6.8 (Compilation-Ready Leakage Function)**.** *Let* $\mathsf{CP}_{\mathtt{m\text{-}evl}}$ *be* $F_{\mathtt{m\text{-}evl}}$*-leaky zero-knowledge. We define the "Compilation-Ready Leakage Function"* F *as the function that on input* $\mathtt{opn}$*-instances leaks no information, while on input* $\mathtt{poly}$*-instances* $\mathbb{x} := (\mathtt{poly}, \mathbb{x}')$ *and witness* $\mathbb{w} := (\vec{p}, \vec{o})$ *does the following:*

1. *Leak* $\{(j, p_j(x_j^{(k)}))\}_{j,k}$
2. *For any* $i \in [m]$ *compute* $\mathbb{x}_{\mathtt{m\text{-}evl},i}$ *from* $\mathbb{x}'$ *and the leaked points (as the honest prover would do), leak points* $F_{\mathtt{m\text{-}evl}}(\mathbb{x}_{\mathtt{m\text{-}evl},i}, (p_j, o_j)_{i \in \mathcal{P}_i})$.

Theorem 3.6.2. *Let* CP *be the CP-SNARK presented above. If* $\mathsf{CP}_{\mathtt{m\text{-}evl}}$ *is* $F_{\mathtt{m\text{-}evl}}$*-leaky zero-knowledge then* CP *is* F*-leaky zero-knowledge (see Definition 3.6.8).*

Proof. We restrict our attention to $\mathtt{poly}$-instances of the form $\mathbb{x} := (\mathtt{poly}, \mathbb{x}_{\mathtt{poly}})$, and with associated witness $\mathbb{w}_{\mathtt{poly}}$, since the $\mathtt{opn}$-instances can be trivially simulated.

We rely on the leaky zero knowledge simulator $\mathcal{S}' = (\mathcal{S}'_0, \mathcal{S}'_1, \mathcal{S}'_2)$ of the scheme $\mathsf{CP}_{\mathtt{m\text{-}evl}}$. In particular, we define the simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1, \mathcal{S}_2)$, where $\mathcal{S}_0$ (resp. $\mathcal{S}_2$) invokes $\mathcal{S}'_0$ (resp. $\mathcal{S}'_2$) on input $\mathbb{x}_{\mathtt{poly}}$. Upon input the statement $\mathbb{x}$ and the leakage $(j, \tilde{y}_{i,j})_{i \in [m], j \in \mathcal{P}_i}, y'_1, \ldots, y'_m \leftarrow F_{\mathtt{poly}}(\mathbb{x}_{\mathtt{poly}}, \mathbb{w}_{\mathtt{poly}})$, $\mathcal{S}_1$ derives, for any i, the statement $\mathbb{x}_{\mathtt{m\text{-}evl},i}$ from $\mathbb{x}_{\mathtt{poly}}$ and $(j, \tilde{y}_{i,j})_{i \in [m]}$, then it runs $\mathcal{S}'_1$ on the derived statement and leakage y'_i. The indistinguishability of the simulated view from the real view can be proved with a hybrid argument where at each step we use the leaky zero-knowledge of $\mathsf{CP}_{\mathtt{m\text{-}evl}}$. □

Theorem 3.6.3. *Let* $\epsilon_{\mathtt{m\text{-}evl}}$ *be the maximum winning probability of a PT adversary against the* $\Phi_{\mathtt{m\text{-}evl}}$*-simulation extractability. Let* $\mathcal{E}$ *be the canonical extractor in the AGM, let* $\mathcal{S}$ *be the* F*-leaky ZK simulators for* CP *(see Definition 3.6.8). For every* $\hat{\Phi} \in \hat{\mathbf{\Phi}}$ *(see Definition 3.6.6), for every adversary* $\mathcal{A}$ *that makes at most* Q_{sim} *simulation queries:*

$$\mathbf{Adv}^{\hat{\Phi}\text{-}se}_{\mathsf{CP},\mathcal{S},\mathcal{E},\mathcal{A}}(\lambda) \leq (Q_{\mathsf{sim}} + 1)\epsilon_{\mathtt{m\text{-}evl}} + \epsilon_{\textit{Aff-MDH}} + \frac{\mathsf{poly}(d,\lambda)}{|\mathbb{F}|}$$

Proof. We consider the *canonical* extractor given by the AGM guarantees, i.e., the extractor that parses and outputs the polynomials in the instance as derived by the coefficients in the algebraic representations.

We start by proving that for any algebraic adversary $\mathcal{A}$ whose forgery satisfies the extractor policy $\mathtt{opn}$-forgery of $\hat{\Phi}$, there exists an algebraic adversary $\mathcal{B}$ whose forgery satisfies the extractor policy $\mathtt{poly}$-forgery of $\hat{\Phi}$.

Lemma 3.6.6. *For any algebraic adversary $\mathcal{A}$ there exists an algebraic adversary $\mathcal{B}$ such that:*

$$\mathbf{Adv}^{\hat{\Phi}\text{-}se}_{\mathsf{CP},\mathcal{A},\mathcal{S},\mathcal{E}}(\lambda) = \mathbf{Adv}^{\hat{\Phi}\text{-}se}_{\mathsf{CP},\mathcal{B},\mathcal{S},\mathcal{E}}(\lambda)$$

Proof. The reduction $\mathcal{B}$ forwards all the simulation queries of $\mathcal{A}$. When $\mathcal{A}$ outputs as a forgery an $\mathtt{opn}$-instance, $\mathcal{B}$ outputs as forgery the instance $\mathbb{x}$ and valid proof π contained in aux_Φ.

We can assume that the representations of the commitments in the $\mathtt{opn}$-forgery and the representations of the same commitments in the auxiliary output aux_Φ are the same; otherwise, if we have two distinct representations for the same commitment, we break the binding of KZG. Thus, the canonical extractor would output the same opening both when extracting the $\mathtt{opn}$-forgery (using the representations in $\mathsf{aux}_\mathcal{E}$) and when extracting from aux_Φ. □

From now on, we parse $\mathbb{x}^*$ as $(\mathtt{poly}, \mathbb{x}^*_{\mathtt{poly}})$, and we define $\pi^*_{\mathtt{poly}} \coloneqq \pi^*$.

We define a hybrid experiment $\mathbf{H}_1$ that is equivalent to the $\hat{\Phi}$-simulation extractability experiment, but additionally the hybrid experiment outputs 0 if the instance $\mathbb{x}^*_{\mathtt{poly}}$ in the forgery of the adversary is in the set of simulation queries and the canonical extractor fails to extract a valid witness.

Lemma 3.6.7. $\mathbf{Adv}^{\hat{\Phi}\text{-}se}_{\mathsf{CP},\mathcal{S},\mathcal{E},\mathcal{A}}(\lambda) \leq \Pr[\boldsymbol{H}_1] + \epsilon_{\mathtt{m\text{-}evl}} + \epsilon_{\textit{Aff-MDH}}$

Proof. The distinguishing event between the original experiment and $\mathbf{H}_1$ is that $\mathcal{A}$ returns a valid "fresh" proof for a statement for which has seen a simulated proof which the extractor cannot extract.

We reduce to $\Phi_{\mathtt{m\text{-}evl}}$-simulation extractability showing an adversary that produces a *fresh* proof for a statement $\mathbb{x}_{\mathtt{m\text{-}evl}}$ for which has seen a simulated proof.

Consider the policy $\Phi_{\mathtt{m\text{-}evl}} = (\bar{\Phi}_0, \bar{\Phi}_1) \in \boldsymbol{\Phi}_{\mathtt{m\text{-}evl}}$ where $\bar{\Phi}_0(\mathsf{pp}_\mathbb{G})$ does the following:

1. Runs the policy $\Phi_0(\mathsf{pp}_\mathbb{G})$ and obtains the set of vectors $\mathcal{Q}_v$.
2. Samples random $\tilde{x}_1, \ldots, \tilde{x}_q$ from $\mathbb{F}$ and computes $\mathcal{Q}_x \coloneqq \{v_j(\tilde{x}_i) : \vec{v} \in \mathcal{Q}_v\}_{i,j}$.
3. Runs for $i \in [Q_{\mathsf{sim}}]$ the sampler $\vec{\mathsf{c}}^{(i)} \leftarrow\!\!\$\ \mathcal{D}(\mathsf{pp}_\mathbb{G})$, where Q_{sim} is the maximum number of simulation queries made by $\mathcal{A}$, and defines $\mathsf{coms} \coloneqq (\vec{\mathsf{c}}^{(i)})_i$.
4. Outputs $(\mathcal{Q}_x, \mathsf{coms})$.

We can assume w.l.g. that $\mathcal{A}$ queries the random oracle on $\mathbb{x}$ before querying the simulation oracle on such an instance.

Consider the adversary $\mathcal{B}$ for the $\Phi_{\mathtt{m\text{-}evl}}$-simulation-extractability game that:

1. Run the adversary $\mathcal{A}$ on parameters ck and $\mathsf{pp}_\Phi \coloneqq (\mathcal{Q}_v, \mathsf{coms})$
2. Parse coms as $(\vec{\mathsf{c}}^{(i)})_{i\in[Q_{\mathsf{sim}}]}$ and keep a list $\mathcal{Q}'_{\mathsf{RO}}$ (initially empty) of the random oracle call of $\mathcal{A}$.
3. At the i-th random-oracle query on input (s, aux), store $(s, \mathsf{aux}, \tilde{x}_i)$ in $\mathcal{Q}'_{\mathsf{RO}}$, and forward $\tilde{x}_i$ to the adversary $\mathcal{A}$.
4. Upon ($\mathtt{poly}$-instance) simulation query with a tuple $(\mathbb{x}_{\mathtt{poly}}, \mathsf{aux}_{\mathtt{poly}})$ from $\mathcal{A}$:
   - Find in $\mathsf{aux}_{\mathtt{poly}}$ the leakage-input for the simulator $((y_{i,j})_{j\in\mathcal{P}_i}, y'_i)_{i\in[m]}$.

- Following the specification of the prover, query the simulation oracle with instances $\mathbb{x}_{\texttt{m-evl},i} = ((\vec{\mathsf{c}}_j)_{j\in\mathcal{P}_i}, x_i^*, (y_{i,j})_{i,j\in\mathcal{P}_i})$, using y_i' as the leakage for the i-th instance. Parse them as a proof for `poly`.

5. Upon forgery $(\mathbb{x}^*_{\texttt{poly}}, \mathsf{aux}^*_{\mathcal{E}}, \pi^* = (\pi^*_{\texttt{m-evl},i}, (y_{i,j})_j)_i)$:
   (a) **Abort** if $\mathcal{A}$ never queried $\mathcal{S}_2$ with $\mathbb{x}^*_{\texttt{poly}}$.
   (b) Let $\tilde{\pi}_{\texttt{poly}} \coloneqq (\tilde{\pi}_{\texttt{m-evl},i}, (\tilde{y}_{i,j})_j)_i$ be the first simulated proof for $\mathbb{x}^*_{\texttt{poly}}$ and let i^* be the index such that either $(y_{i^*,j}) \neq (\tilde{y}_{i^*,j})$ or $\tilde{\pi}_{\texttt{m-evl},i} \neq \pi_{\texttt{m-evl},i}$, return the forgery $(\mathbb{x}^*_{\texttt{m-evl},i^*}, \mathsf{aux}^*_{\mathcal{E}}, \pi^*_{\texttt{m-evl},i^*})$.

We need to show that if $\hat{\Phi}$ holds for $\mathcal{A}$ then the policy $\Phi_{\texttt{m-evl}}$ holds for $\mathcal{B}$. Notice that, by condition (1) of the semi-adaptive simulation queries property of $\hat{\Phi}$ and the definition of Φ_0 the reduction $\mathcal{B}$ calls its own simulator on points in $\mathcal{Q}_x$, moreover conditions (2,3) of the semi-adaptive simulation queries property of $\hat{\Phi}$ easily imply respectively the Commitment Check and the Algebraic Consistency of $\Phi_{\texttt{m-evl}}$ (cf. Definition 3.5.4).

By inspection on the forgery, if the reduction $\mathcal{B}$ does not abort and there exists only one $\tilde{\pi}_{\texttt{poly}}$ associated with the forged instance then $\mathcal{B}$ matches the predicate $\Phi^{\text{der}}_{\mathsf{ext}}$ of the $\Phi_{\texttt{m-evl}}$ policy (cf. Definition 3.5.5). We show that, because of condition (2) and the algebraic consistency check of the semi-adaptive simulation queries property of $\hat{\Phi}$, there exists indeed only one simulated proof $\tilde{\pi}_{\texttt{poly}}$ for each queried instance. In fact, from an adversary that asks twice the same instance to the simulator, let say at query i and i', we can derive that $\vec{C}^{(i)}\cdot\vec{\mathsf{c}}^{(i)} + \vec{F}^{(i)}\cdot\mathsf{ck} = \vec{C}^{(i')}\cdot\vec{\mathsf{c}}^{(i')} + \vec{F}^{(i')}\cdot\mathsf{ck}$. If the matrices $\vec{C}^{(i)}$ or $\vec{C}^{(i')}$ are non-zero then we can break the $\mathcal{D}$-Aff-MDDH assumption. On the other hand, if both matrices are the zero matrix, we have that the commitments are fully defined given the ck and thus by the algebraic consistency check the leakage for the simulated proofs must be the same (and thus the proofs are the same). □

The next hybrid experiment $\mathbf{H}_2$ is equivalent to $\mathbf{H}_1$, but additionally it outputs 0 if the forgery of the adversary is valid and the canonical extractor fails to extract valid witnesses for the `m-evl`-instances derived by $\mathbb{x}^*_{\texttt{poly}}$.

Lemma 3.6.8. $\Pr[\boldsymbol{H}_1] \leq \Pr[\boldsymbol{H}_2] + \frac{\epsilon_{\texttt{m-evl}}}{Q_{\mathsf{sim}}}$

Proof. We reduce again to $\Phi_{\texttt{m-evl}}$-simulation-extractability We consider the same $\bar{\Phi}_0(\mathsf{pp}_{\mathbb{G}})$ as in the previous lemma.

Consider the adversary $\mathcal{B}$ that:

1. Sample an index $q^* \leftarrow\!\!\$\ [Q_{\mathsf{sim}}]$.
2. Run the adversary $\mathcal{A}$ on parameters ck and $\mathsf{pp}_\Phi \coloneqq (\mathcal{Q}_v, \mathsf{coms})$
3. Parse coms as $(\vec{\mathsf{c}}^{(i)})_{i\in[Q_{\mathsf{sim}}]}$ and keep a list $\mathcal{Q}'_{\mathsf{RO}}$ (initially empty) of the random oracle call of $\mathcal{A}$.
4. At the i-th random-oracle query on input (s, aux):
   - if $i \neq q^*$ then store $(s, \mathsf{aux}, \tilde{x}_i)$ in $\mathcal{Q}'_{\mathsf{RO}}$, and forward $\tilde{x}_i$ to $\mathcal{A}$

- else parse s as $(\mathtt{msg}, \vec{\mathsf{c}}, (G^{(k)}, \vec{v}^{(k)})_k)$, find k^* such that $\forall j : \mathsf{deg}(v_j^{(k^*)}) \geq 1$, set $\mathsf{aux}' := (\mathsf{aux}, (v_j^{(k^*)})_{j\in[n]})$, and send the query (s, aux') to $\mathcal{S}_2$, and forward a to the adversary $\mathcal{A}$

5. Upon simulation query with a tuple $(\mathbb{x}_{\mathtt{poly}}, \mathsf{aux}_{\mathtt{poly}})$ from $\mathcal{A}$ (as in the previous lemma):
   - find in $\mathcal{Q}'_{\mathsf{RO}}$ the tuple $(\mathbb{x}_{\mathtt{poly}}, \cdot, \tilde{x})$
   - find in $\mathsf{aux}_{\mathtt{poly}}$ the leakage-input for the simulator $((y_{i,j})_{j\in\mathcal{P}_i}, (y'_i))_{i\in[m]}$
   - following the specification of the prover, query the simulation oracle with instances $\mathbb{x}_{\mathtt{m\text{-}evl},i} = ((\vec{\mathsf{c}}_j)_{j\in\mathcal{P}_i}, x_i^*, (y_{i,j})_{i,j\in\mathcal{P}_i})$, using y'_i as the leakage for the i-th instance
6. Upon forgery $(\mathbb{x}^*_{\mathtt{poly}}, \mathsf{aux}^*_{\mathcal{E}}, \pi^* = (\pi^*_{\mathtt{m\text{-}evl},i}, (y_{i,j})_j)_i)$
   (a) **abort** if $\mathbb{x}^*_{\mathtt{poly}}$ was not queried at the q^*-th random oracle query by $\mathcal{A}$
   (b) **abort** if $\forall j \in [n]$ c_j can be extracted as $p_j(X)$ and $\forall i : p_j(x_i^*) = y_{i,j}$,
   (c) Let j^*, i be such that $p_{j^*}(x_i^*)$ the previous check does not hold. Return the forgery $(\mathbb{x}^*_{\mathtt{m\text{-}evl},i^*}, \mathsf{aux}^*_{\mathcal{E}}, \pi^*_{\mathtt{m\text{-}evl},i^*})$.

Let Abort be the event that $\mathcal{B}$ aborts. We notice that $\mathcal{B}$ could abort if one of two distinct events happens. The first event Abort_1 in Item 6a, and the second event Abort_2 is the condition in Item 6b. Notice that the distinguishing event between the two hybrids implies that Abort_2 does not happen. Moreover, notice that when the reduction does not abort, it returns a valid proof that either the canonical extractor cannot extract or such that the extracted polynomial $p_j(x_i^*) \neq y_{i^*,j^*}$; thus, if the policy $\Phi_{\mathtt{m\text{-}evl}}$ is valid the reduction wins the $\Phi_{\mathtt{m\text{-}evl}}$-simulation extractability experiment. Namely,

$$\Pr\left[\bar{\Phi}_1 \wedge \neg\mathsf{Abort} | b = 0\right] \leq \epsilon_{\mathtt{m\text{-}evl}}$$

The event Abort_1 only depends on the uniformly random index q^* which is independent of the view of the adversary $\mathcal{A}$, i.e., $\Pr[\neg\mathsf{Abort}_1] = \frac{1}{Q_{\mathsf{sim}}}$. Notice that $\mathcal{A}$ wins the $\Phi_{\mathtt{poly}}$-simulation extractability experiment when $\neg\mathsf{Abort}_2$ happens and Φ_1 holds. Thus, we only need to show that, conditioned on $\neg\mathsf{Abort}$, when the policy Φ_1 holds w.r.t. the view of $\mathcal{A}$ then the policy $\bar{\Phi}_1$ holds w.r.t. the view of $\mathcal{B}$. By construction of $\mathcal{Q}_x$ and the answers to the random oracle queries to $\mathcal{A}$, if the simulation query of $\mathcal{A}$ has $\vec{v}^{(k)} \in \mathcal{Q}_v$ then the evaluation points derived are in $\mathcal{Q}_x$. Moreover, it is not hard to see that the consistency check of the two policies match: in particular, the conditions on matrix $\vec{C}_i$ (resp $\vec{F}_i$) are equivalent to the consistent opening check described in Item 3 of the extraction policy (the adversary is considered valid if its queries form a solvable linear system of equations defined by the coefficients of the linear combinations of commitments and the evaluation values). As for the forgery, in case of $\neg\mathsf{Abort}_1$, the entry $(\mathbb{x}^*_{\mathtt{poly}}, \mathsf{aux}', a^*)$ is indeed in the list of random oracle checked by the $\mathtt{m\text{-}evl}$-policy: in fact, it is the only query. And, by construction, aux' contains the polynomial $v_{j^*}^{(k^*)}$. Putting things together we have the statement of the lemma. □

The next hybrid experiment $\mathbf{H}_3$ is equivalent to $\mathbf{H}_2$, but additionally it outputs 0 if the instance in the forgery of the adversary is valid and the canonical extractor fails to extract valid witness, namely:

$$\exists k : \bar{G}^{(k)}(X, p_1(v_1^{(k)}(X)), \ldots, p_n(v_n^{(k)}(X))) \not\equiv 0$$

Lemma 3.6.9. $\Pr[\boldsymbol{H}_3] \leq \frac{\mathsf{poly}(d,\lambda)}{|\mathbb{F}|}$.

Proof. First notice that by the changes introduced in the previous hybrid the canonical extractor must extract valid polynomials and $p_j(x_i^*) = y_{i,j}$ for any i and $j \in \mathcal{P}_i$. Also, x^* is sampled after all the polynomials extracted by the canonical extractor are defined.

Let d be the degree of $\bar{G}^{(k)}(X, p_1(v_n^{(k)}(X)), \ldots, p_n(v_n^{(k)}(X)))$; such value is polynomial in the security parameter and the maximum degree for the commitments. If the verification passes we have:

$$\bar{G}^{(k)}(x^*, p_1(v_1^{(k)}(x^*)), \ldots, p_n(v_n^{(k)}(x^*))) = 0$$

which, by Schwartz-Zippel lemma happens with probability $\frac{d}{\mathbb{F}}$. □

□

Chapter 4

Non-malleable Real-World zkSNARKs

This chapter is extracted from "Real-world Universal zkSNARKs are non-malleable", published in CCS 2024.

4.1 Introduction

In Section 1.3.1 we have seen that several recent results [GOP+22, DG23, GKK+22, FFK+23, KPT23] prove the simulation extractability of existing zkSNARKs, e.g., Bulletproofs [BBB+18], Spartan [DG23], Sonic [MBKM19], PLONK [GWC19], Marlin [CHM+20], Lunar [CFF+21] and Basilisk [RZ21]. While some of these works, such as [GOP+22, DG23, GKK+22], prove the SE of specific zkSNARKs, the approach taken by us in Chapter 3 and in a concurrent work by Kohlweiss, Pancholi and Takahashi [KPT23] is to prove the simulation extractability of a broad class of zkSNARKs, namely those built via the popular approach combining polynomial interactive oracle proofs (PIOPs) and polynomial commitments. Furthermore, we have seen that the works in the latter category are of particular interest as they give an SE recipe that is generic, and thus it can benefit both existing and future schemes.

Given this state of the art, one may therefore ask if there is more to know about the SE of universal zkSNARKs based on PIOPs. However, a closer look at the recent results reveals *two important gaps that do not allow concluding that the "real world" versions of schemes like PLONK and Marlin are simulation extractable.*

Theory vs. Implementation. The first gap lies in that the versions of these schemes that offer the best performance and are eventually implemented in software libraries[1] slightly depart from the ones obtained through the PIOP-to-zkSNARK vanilla compilation. The difference is in the last step. In order to maximize efficiency, they apply an optimization (that we call *linearization trick*, also known as Maller's optimization) [GWC19, OL] that leverages the homomorphic properties of the KZG polynomial commitment to reduce the number of field elements in the proof. This optimization though changes the zkSNARK verification algorithm in a way that escapes the SE security analysis we do in Chapter 3; a similar limitation holds for the work of [KPT23].

Delegation phase. The second gap is that both the aforementioned frameworks capture PIOPs in which some polynomials are evaluated on a random challenge chosen in the last

[1] E.g., https://github.com/dusk-network/plonk, https://github.com/arkworks-rs/marlin

round. This is however not the case for Marlin and Lunar in which polynomials involving the witness are evaluated on a challenge chosen before the last round, which is witness-independent and needed only for verifier's efficiency (what we call a *delegation phase*). For this reason, up to now we can only argue the SE of small variants of Marlin and Lunar.

4.1.1 Our contributions

In this chapter, we resolve the two limitations above, and we give the first proof of SE of the "real world" optimized versions of zkSNARKs which include PLONK, Marlin, Lunar, and Basilisk. To achieve these results, we improve the techniques presented in Chapter 3 in several ways:

(1) we formalize the compilation recipe based on the linearization trick, and we prove that, under a set of minimal constraints, PIOPs can be compiled to SE zkSNARKs using the linearization trick optimization;

(2) we refine the set of conditions to compile a PIOP to a zkSNARK, notably removing the artificial one from Chapter 3 that prevented capturing Marlin and Lunar, and thus we broaden the class of PIOPs that can be compiled in an SE manner;

(3) we simplify and generalize the conditions under which KZG can be proved simulation-extractable.

As a byproduct of (1) and (3), we obtain the first security analysis of the linearization trick optimization. We show potentially insecure instantiations as well as a characterization of the conditions that make it secure even in terms of plain knowledge-soundness, in the AGM with oblivious sampling (AGMOS) [LPS23].

Some of our definitions and techniques to prove SE may appear rather convoluted. We would like to note that this is due to the wish of capturing SE for *existing* protocols, without introducing any change, which is a challenging goal. As an example, in Chapter 3 we gave a simple condition to safely compile a PIOP: that a witness-dependent polynomial is evaluated on a random challenge chosen in the last round. However, this condition is not met by some protocols, which in this work we eventually prove to be SE. This required us to elaborate more complex conditions to explain why this is possible.

4.1.2 Organization of the chapter

We provide a more comprehensive explanation of our results in Section 4.2. In Section 4.4 we introduce a new computational assumption that is used to prove some of our results. In Section 4.5 we extend the framework of policy-based simulation extractability of Section 3.4, we formalize the linearization trick, and we prove its simulation extractability in the AGM. Finally, we revisit the compiler from PIOPs to zkSNARKs, and we show that our compiler effectively captures the optimized versions of popular schemes, such as Plonk and Marlin.

4.2 A Technical Overview of Our Results

4.2.1 Revisiting PIOP-based zkSNARKs

As discussed in Section 2.4.3.3, a common approach to design zkSNARKs is to first construct an information-theoretic protocol that achieves the desired functionality in an idealized model and then remove the idealized component by compiling it into a zkSNARK via the use of a computationally-secure primitive [Ish19, Ish20].

For the information-theoretic part, the most popular instantiation uses PIOPs [GWC19, BFS20, CHM+20, Sze20, CFF+21]. However, the kind of queries that the verifier does may vary: the simplest form of queries is the evaluation of polynomials (cf. [CHM+20, BFS20]), but other models (cf. [GWC19, CFF+21]) consider more general queries that state the validity of polynomial equations over a subset of the committed polynomials.

Our generalization: $\mathcal{R}$-PIOP. To keep all these different notions of PIOP under the same umbrella, in our work, we define the notion of $\mathcal{R}$-PIOP where the verifier's queries are instances belonging to the *oracle* relation $\mathcal{R}$. Roughly speaking, an oracle relation is an NP-relation where the instances can refer to polynomial oracles [CBBZ23]. Under this definition, Marlin uses an $\mathcal{R}_{\texttt{evl}}$-PIOP, where $\mathcal{R}_{\texttt{evl}}$ is the relation that checks that a polynomial oracle evaluates to y at point x, while PLONK [GWC19] and Lunar [CFF+21] are $\mathcal{R}_{\texttt{poly}}$-PIOPs, where $\mathcal{R}_{\texttt{poly}}$ is the relation that checks polynomial equations over polynomial oracles.

How to compile $\mathcal{R}$-PIOPs? Unfortunately, zkSNARKs obtained by (mechanically) applying compilations from $\mathcal{R}_{\texttt{evl}}$-PIOPs are often suboptimal proof systems, due to the fact that one should include in the proof a field element for each evaluation of a polynomial oracle. In particular, such compilations cannot leverage on the homomorphic property that many polynomial commitments, such as KZG [KZG10], have. Thus, subsequent optimizations usually accompany, and slightly change, the formally analyzed zkSNARKs. Instead, zkSNARKs compiled from $\mathcal{R}_{\texttt{poly}}$-PIOPs can defer all the optimizations to the richer and more expressive (sub)-proof systems for $\mathcal{R}_{\texttt{poly}}$.[2] Yet, in practice, the latter proof systems are often reduced to the former via a random point evaluation.

One of the most common optimizations, based on homomorphic commitments, is the so-called *linearization trick*, sometimes referred as the Mary Maller's optimization [GWC19, OL]. This optimization allows reducing the number of field elements in the final proofs. For example, to prove that $A(x)B(x)+C(x)=y$ holds for committed polynomials A,B,C, and values x and y, one can prove that $B(x)=y_b$ for some y_b, the verifier uses the homomorphism of the polynomial commitment to obtain the commitment to the *linearization* polynomial $L(X) := A(X)y_b + C(X)$, and then the prover proves that $L(x)=y$, saving from naively evaluating all the polynomials on x.

Building on this idea, PLONK [GWC19] describes a general recipe to compile an $\mathcal{R}_{\texttt{poly}}$-PIOP to zkSNARK. The procedure first finds the minimal sub-set of polynomials that one should evaluate in order to generate the linearization polynomial, and then it (batch) evaluates all the polynomials in this subset and the linearization polynomial on a fresh random point.[3]

[2] On the downside, PIOPs based on polynomial equations, while at an informal level are easier to describe, tend to have harder-to-parse full specifications.

[3] Such a random point is needed to reduce polynomial identity tests into equations over field elements, through the Schwartz-Zippel Lemma.

On the (in)security of the linearization trick. It turns out that this general recipe is not always sound. In fact, the work of [LPS23] shows a counter-example to the extractability of the linearization trick when using the KZG polynomial commitment. In particular, assume the adversary does not know the representation of a group element c (using the lingo of [LPS23], c is an obliviously sampled element), and sets the three polynomial commitments of the example above as $(\mathsf{c}_A, \mathsf{c}_B, \mathsf{c}_C) = (\mathsf{c}, [b]_1, -b \cdot \mathsf{c})$. According to [LPS23], only the commitment c_B is extractable in the algebraic group model, namely the adversary can give an algebraic representation (under the basis of the elements of the SRS) only for c_B. The linearization commitment would be equal to $\mathsf{c}_L = \mathsf{c}b - b\mathsf{c} = [0]_1$, which is independent of the evaluation point x. The adversary can clearly provide an evaluation proof at x for c_L, in spite of not knowing the polynomials implicitly committed in c_A and c_C. In particular, this counter-example shows that we can only extract the second of the three polynomials, under the (more realistic) algebraic group model where the adversary gets to see group elements, besides the SRS, for which it does not know their algebraic representations.

Here we generalize the attack of [LPS23] by considering the general case where, for committed polynomials $(A_i, B_i)_{i\in[n]}$, we want to prove that $\sum_i A_i(x)B_i(x) = y$. In particular, we let $\mathcal{R}_{\mathsf{lin}}$ be the relation where the instances are tuples of the form $((\mathsf{a}_i, \mathsf{b}_i)_{i\in[n]}, x, y)$ such that for field elements x and y, and any i, a_i (resp. b_i) is a commitment to A_i (resp. B_i) for which the equation above holds. What we call (the zkSNARK for) the linearization trick for KZG is the proof system that proves $y_i = A_i(x)$ for any i and then, using the homomorphic property of KZG, generates the linearization commitment $\mathsf{c}_L = \sum_i y_i \cdot \mathsf{b}_i$, and proves $L(x) = \sum_i y_i B_i(x) = y$.

Our generic attack works whenever the polynomials A_i are linearly dependent. The attacker can set, for example, the commitments for the polynomials B_i to $\mathsf{b}_i = \alpha_i \cdot \mathsf{c}$, for an obliviously sampled group element c and for carefully chosen values $(\alpha_i)_i$ such that $\sum_i \alpha_i A_i(X) = 0$. It is easy to see that this adversary can generate a proof for $\sum_i A_i(x^*)B_i(x^*) = 0$, for any x^*, without knowing all the polynomials B_i.

We do not formalize this attack further as we use it mainly as a motivation for our constructive results. Indeed, we use the intuition behind this counter-example in order to show that the independence of the polynomials A_i is the missing piece of the puzzle to prove extractability of (the zkSNARK defined from) the linearization trick. In particular, the correct recipe for the general compiler proposed by [GWC19] should look not only for the sub-set that minimizes the number of polynomials to open, but should also make sure that the polynomials in such a sub-set are linearly independent. Luckily, the linear independence holds for the subset of polynomials chosen for this optimization in PLONK.[4] We obtain a formalization of this security claim as a corollary of our results on the SE of KZG (see next section).

To summarize, our first set of results deals with proving that the linearization trick for KZG is, under certain conditions, simulation-extractable. We do this in two main steps. First, we consider the SE of KZG evaluation proofs in which the commitment is obtained by a linear combination of other commitments (cf. Sections 4.2.2 and 4.5.1.1). Second, we analyze the sufficient conditions on the A_i polynomials that make the linearization trick simulation extractable (cf. Sections 4.2.3 and 4.5.2).

[4]Here we are simplifying: the verification in PLONK uses the linearization trick on a mix of polynomials that comes both from the prover and the indexer (i.e., polynomials committed in the specialized reference string). Indexer's polynomials are trivially extractable as they are part of the statement, thus we can refine the property of linear independence by focusing on the polynomials that are not *coupled* with indexer's ones.

4.2.2 Simulation Extractability of KZG for linearized commitments

In Chapter 3 we introduced the notion of policy-based SE, that, roughly speaking, ensures that a zkSNARK is simulation-extractable whenever the adversary complies with a pre-defined policy. This generalized notion of SE is convenient (and necessary) to formalize the simulation-extractable properties of malleable schemes such as KZG.

We summarize the security game of SE for KZG in the algebraic group model. The adversary obtains a list of obliviously-sampled commitments $\mathsf{c}_1, \ldots, \mathsf{c}_n$ where $\mathsf{c}_i \in \mathbb{G}_1$, and it has oracle access to a simulation oracle that, upon input tuples (c, x, y), outputs simulated proofs $\pi = (\mathsf{c} - [y]_1)(s-x)^{-1}$. Additionally, the adversary has oracle access to a random oracle.[5] Eventually, the adversary outputs its forgery π^* for an instance (c^*, x^*, y^*). Standard simulation extractability would just require that the instance was never queried to the simulation oracle. Additionally, the policy we define in Section 3.5 requires that:

1. The queries of the adversary do not create an algebraic inconsistency in terms of the proved statements. For example, the adversary cannot obtain simulated proofs for (c, x, y) and (c, x, y') with $y \neq y'$. This constraint is strictly necessary to prove SE for KZG.

2. The evaluation points x for the simulation queries belong to an arbitrary, but fixed ahead of time, set $\mathcal{Q}_x$. This property is called semi-adaptive queries.

3. The group elements c asked in the simulation queries could not be (algebraically) derived using previously obtained simulated proofs.

4. The forgery's evaluation point x^* must be random and independent of c^*. To enforce this, we check that x^* is derived by applying the random oracle to a string that contains an encoding of c^*.

In this chapter, we substantially simplify the policy above by removing the second and third constraints. Besides providing a cleaner and simpler notion of security, removing these constraints has two extra benefits: Removing the second constraint allows proving the PIOP-to-zkSNARK compiler secure in the *non-programmable* random oracle model; removing the third constraint allows extending the PIOP-to-zkSNARK compiler to work with *commit-and-prove* relations (namely, the relation proved by the zkSNARK can have commitments as part of the instance). One limitation of our technique to remove the second constraint is that we need to make a stronger cryptographic assumption than the q-SDH assumption that we call *one-more* q-SDH assumption. This assumption additionally provides an oracle that can be adaptively queried on field element x and (small) natural number i and returns $[(s-x)^{-i}]_1$. We show that the *one-more* q-SDH assumption holds in the generic group model and that we can reduce a non-adaptive version of the *one-more* SDH to the plain q-SDH assumption.

Moreover, we generalize the fourth constraint. Specifically, we change the constraint by allowing c^* to be a commitment to a linearization polynomial. To do so we check that x^* is derived from the random oracle with inputs commitments $(\mathsf{b}_i)_i$ and polynomials[6] $(A_i)_i$ and that $\mathsf{c}^* = \sum A_i(x^*)\mathsf{b}_i$. Proving SE using the latter generalization turns out to be the necessary

[5] This is not strictly necessary, and it could be modeled differently. However, it is a convenient model since the PIOP-to-zkSNARK compiler uses the Fiat-Shamir transformation.

[6] Technically, we treat these latter polynomials as auxiliary information that the adversary must "declare" before seeing x^*.

heavy lifting to perform in order to, then, show that the linearization trick is (policy-based) simulation-extractable, as summarized in the following theorem.

Informal Statement of Theorem 4.5.1. *The evaluation proof of KZG polynomial commitment is (policy-based) simulation-extractable in the algebraic group model under our simplified and generalized policy where the forgery can contain a commitment to a linearization polynomial.* The obliviously-sampled commitments $\mathsf{c}_1, \ldots, \mathsf{c}_n$ not only allow to give an interesting notion of SE in the algebraic group model, they also naturally extend our model to include the algebraic group model with obliviously-sampled elements, as considered in [LPS23]. For this reason, by proving SE of the linearization trick for KZG w.r.t. this new set of constraints, we can derive the following Corollary by considering the subclass of adversaries that do not query the simulation oracle.

Corollary 4.2.1. *PLONK and Marlin are knowledge-sound in the AGMOS.*

4.2.3 Simulation extractability of the linearization trick

Unfortunately, when the adversary can make simulation oracle calls, the condition of linear independence, sufficient and necessary to restore the (plain) knowledge extractability of the linearization trick, is not sufficient. In fact, consider an adversary, holding an obliviously sampled group element c, that asks for a simulated proof on $(\mathsf{c}, 0, 0)$, namely a proof that the polynomial committed in the *commitment* c evaluates to 0 at point 0. Let $\pi = \mathsf{c}/s$ be the simulated proof. The adversary can generate an instance for $\mathcal{R}_{\mathsf{lin}}$ that is not extractable even if the polynomials A_i are linearly independent. It could set the polynomials $A_1(X) = 1$ and $A_2(X) = -X$, thus $(\mathsf{c}_{A_1}, \mathsf{c}_{A_2}) = ([1]_1, [-s]_1)$, and then sets $(\mathsf{c}_{B_1}, \mathsf{c}_{B_2}) = (\mathsf{c}, \pi)$ as the (unextractable) commitments to the polynomials B_1 and B_2 respectively. Now, for any arbitrary x^* it can generate a forgery, by setting the forged proof π^* to c/s. Its validity follows from the fact that $1 \cdot \mathsf{c} - x^*\mathsf{c}/s = (s - x^*)\mathsf{c}/s$. The reason why this attack works is that KZG proofs and KZG commitments belong to the same domain; thus a proof can be reinterpreted as a commitment. Through this lens, the simulation oracle allows the adversary to push down *the degree of* the unextractable polynomials. Looking at the counter example from the perspective of formal polynomials, we have that:

$$\sum_{i=1,2} A_i(X)B_i(X) = A_1(X) \cdot \mathsf{c} + A_2(X) \cdot \tfrac{\mathsf{c}}{X} = A_1(X) \cdot \mathsf{c} + \tfrac{A_2(X)}{X} \cdot \mathsf{c} = 0.$$

The problem is that, while A_1 and A_2 are linearly independent, the polynomials $A_1(X)$ and $A_2(X)/X$ are not so.

As our technical contribution, we show that *higher-degree* linear independence between the polynomials A_i is necessary and sufficient to obtain SE of the linearization trick for the KZG commitment scheme. In particular, instead of defining independence of the polynomials A_i as the condition $\sum \alpha_i A_i \neq 0$ for any choice of $\alpha_i \in \mathbb{F}$, we define their independence w.r.t. any of $\alpha_i \in \mathbb{F}_{\leq \nu}[X]$, for a parameter $\nu \in \mathbb{N}$.

It remains to understand how to set such a parameter ν. To do so, we define a notion of level for *proofs of proofs* that, roughly speaking, indicates how many times the adversary sequentially queried the simulation oracle on an obliviously sampled group element or elements algebraically derived from it. For example, the level of an obliviously sampled element is zero, the level of a proof on it is one, and the level of a proof of a proof could possibly increase to

two if we queried twice on the same evaluation point, or remain the same otherwise[7], and so on.

Informal Statement of Theorem 4.5.2. *The linearization trick for KZG polynomial commitment is (policy-based) simulation-extractable in the AGM under our simplified and generalized policy and assuming that the extracted polynomials $A_1, \ldots, A_n$ are independent for a parameter ν and the maximum level reached by simulated proofs queried by the adversary is smaller or equal to ν.*

The above theorem completes the set of results that we need to obtain SE when instantiating the PIOP-to-zkSNARK compiler with KZG. Next, we address the SE requirements at the PIOP level.

4.2.4 Capturing PIOPs with delegation phase

The work of [FFK+23] showed that only a subset of all the PIOPs can be compiled to SE zkSNARKs. For example, if we take a PIOP for the product relation $\mathcal{R} \times \mathcal{R}$ which, internally, sequentially runs two instances of a PIOP for $\mathcal{R}$, we can incur copy-paste attacks that re-use a simulated proof for the first instance in $\mathcal{R}$ and honestly prove the knowledge for the second instance. To avoid these pathological cases, [FFK+23] introduced the notion of *compilation-safeness* that gives a sufficient condition for a PIOP to be compiled to SE zkSNARK. In a nutshell, a PIOP is compilation-safe if it has a *witness-dependent* last round. Here, by "witness-dependent round", we mean that the polynomials sent at such a round store information that depends on the witness and are necessary to extract the full witness at the PIOP level.

However, Marlin [CHM+20] and other proof systems [CFF+21, RZ21] based on Checkable Subspace Sampling Arguments [RZ21] are not compilation-safe. They have a two-phase algorithm for the prover where the first phase is witness-dependent, while the second phase, which we call *delegation phase*, is witness-independent, and in particular is performed to enable succinct verification. For PIOPs with delegation, we need a more careful compilation strategy. To avoid copy-paste attacks that would copy the witness-dependent transcript from a simulated proof and compute a fresh witness-independent suffix for the forgery, we need to make sure that (1) the polynomial oracles sent during the delegation phase are committed using a deterministic commitment and that (2) the delegation phase is unique at the PIOP level, namely, there is only one possible answer that any malicious prover for the PIOP can send in the delegation phase, once fixed the messages of all the previous rounds[8]. With this characterization of PIOPs we can prove the following theorem:

Informal Statement of Theorem 4.7.1. *PIOPs with delegation phase can be compiled to simulation-extractable commit-and-prove zkSNARKs with the linearization trick optimization. Also, security holds in the algebraic group model with oblivious sampling and in the non-programmable random oracle assuming the one-more q-SDH assumption.*

In the informal theorem above we swept under the rug many details. In particular, the reader may wonder about the connection of the independence parameter ν for the security of the linearization trick and the requirements for the compilation above. What we show in the

[7]Briefly, the reason why the level does not increase in this case is because the rational function $1/((X - x_1)(X - x_2))$ is in the linear span of $(X - x_1)^{-1}$ and $(X - x_2)^{-1}$.

[8]For example, Marlin compiled using hiding KZG, or with FRI-based polynomial commitments, is provably not *strong* simulation extractable, while it could still be proved *weak* simulation extractable.

proof of the compiler is that the parameter ν can be kept very low. In fact, the reduction from SE zkSNARK to the SE of the linearization trick (obliviously) samples fresh commitments for each simulation query made by the adversary and, assuming that the PIOP is zero-knowledge, the reduction needs to query (linearization trick) simulated proofs only for this fresh batch of commitments, thus bounding the level of proof of proofs to, at maximum, the number of evaluations needed by a single execution of the PIOP.

In Section 4.6.4 we show that PLONK and Marlin have PIOPs fulfilling our requirements. Combining this with the theorem above, we obtain our main results on the SE of these schemes.

4.3 Preliminaries

We start by defining the notion of ν-*independent* polynomials.

Definition 4.3.1. *Let $\mathcal{A} = \{A_i\}_{i\in[n]}$ be a set of polynomials in $\mathbb{F}[X]$ and $\nu \in \mathbb{N}$. We say that $\mathcal{A}$ are ν-independent polynomials if $\forall(\alpha_j)_j \in \mathbb{F}_{\leq\nu}[X]$: $\sum_j \alpha_j A_j(X) \neq 0$.*

Lemma 4.3.1. *Let $J \in \mathbb{N}$, and let $\{x_j\}_{j\in[J]} \subset \mathbb{F}_q$, $(\nu_j)_{j\in[J]} \in \mathbb{N}^J$, $\nu^* = \sum_j \nu_j$ and let $S \coloneqq \mathsf{Span}(\{1\} \cup \left\{(X - x_j)^{-k}\right\}_{j\in[J],k\in[\nu_j]})$. Consider the function ϕ with domain S that maps rational functions Ω to polynomials $\Omega \cdot \prod_j (X - x_j)^{\nu_j}$. The function ψ maps is a morphism with image the set $\mathbb{F}_{\leq\nu^*}[X]$.*

Proof. First, notice that any element $v \in S$ can be written as a linear combination of the form $\alpha + \sum_{j,k} \beta_{j,k}(X - x_j)^{-k}$. It holds that ψ carries over the basis since $\psi(v) = \psi(\alpha + \sum_{j,k} \beta_{j,k}(X - x_j)^{-k}) = \alpha\psi(1) + \sum_{j,k} \beta_{j,k}\psi((X - x_j)^{-k})$. The only thing left to prove is that $\psi(1) \cup \psi((X - x_j)^{-k})$ is indeed a basis for $\mathbb{F}_{\leq\nu^*}[X]$. We notice that of $\psi((X - x_j)^{-k})$ is of the form $n_{j,k}(X) \coloneqq (X - x_j)^{\nu_j - k} \prod_{j'\neq j}(X - x'_j)^{\nu_{j'}}$, and $\psi(1)$ is simply $n(X) \coloneqq \prod_j (X - x_j)^{\nu_j}$. To conclude the proof, we show that these polynomials are linearly independent. Given the fact that they are $\nu^* + 1$ polynomials of degree at most ν^*, this is equivalent to prove that they span $\mathbb{F}_{\leq\nu^*}[X]$.

Let $f(X) \coloneqq \alpha n(X) + \sum_{j,k} \alpha_{j,k} n_{j,k}(X)$. We need to prove that $f(X) \equiv 0$ if and only if $\alpha = \alpha_{j,k} = 0$. For all j, it must be that $f(x_j) = 0$. We have that for any x_j: $f(x_j) = \alpha n(x_j) + \sum_{j,k} \alpha_{j,k} n_{j,k}(x_j) = \alpha_{j,\nu_j} \prod_{j'\neq j}(x_j - x_{j'})$, which is equal to 0 if and only if $\alpha_{j,\nu_j} = 0$. We can rewrite $f(X)$ as:

$$f(X) = \alpha n(X) + \sum_{j,k\leq\nu_j} \alpha_{j,k\leq\nu_j} n_{j,k}(X) = f'(X) \prod_j (X - x_j)$$

where $f'(X)$ is equal to:

$$\alpha \prod_j (X - x_j)^{\nu_j - 1} + \sum_{j,k} \alpha_{j,k} (X - x_j)^{\nu_j - k - 1} \prod_{j'} (X - x_{j'})^{\nu_{j'} - 1}$$

Note that if $f(X) \equiv 0$, also $f'(X) \equiv 0$. We can recursively apply the same argument, proving that all the coefficients $\alpha_{j,k} = 0$, for all $k > 0$. In the final step, we can write $f(X)$ as $\alpha \prod_j (X - x_j)$ which is equal to 0 if and only if $\alpha = 0$. □

RO transcript. In our work, we often need to enforce that a point x is random and independent w.r.t. a bunch of elements. To capture this scenario, we check that x is derived

by applying the random oracle (RO) to a string that either (i) contains an encoding of the elements or (ii) the output of another RO query that satisfies the first condition, and so on. We use the shortcut $(x_1, \dots x_n; y_1, \dots, y_m) \to_{\mathsf{RO}} a$ to indicate that there is a list of tuples $(s_1, \mathsf{aux}_1), \dots, (s_k, \mathsf{aux}_k)$ and a list $(a_i)_{i\in[k-1]}$ such that

1. $\forall i \in [k-1] : \mathsf{RO}(s_i, \mathsf{aux}_i) = a_i$, and $\mathsf{RO}(s_k, \mathsf{aux}_k) = a$
2. $\forall i \in [k-1] : a_i$ is a substring of s_{i+1}
3. $\forall j \in [n], \exists i \in [k] : x_j$ is a substring of s_i
4. $\forall j \in [m], \exists i \in [k] : y_j$ is contained in aux_i

4.4 The OMSDH Assumption

Definition 4.4.1. *The* non-adaptive *n-one-more d-strong DH assumption holds for a bilinear group generator* $\mathsf{GroupGen}$ *if for any set $\mathcal{Q}_x$ of cardinality less or equal to n, and for every PPT adversary whose queried points belongs to $\mathcal{Q}_x$, namely, for the class of adversaries whose query points are chosen independently of the randomness of the experiment, the advantage $\mathbf{Adv}^{(n,d)\text{-}\mathtt{OMSDH}}_{\mathsf{GroupGen},\mathcal{A}}(\lambda)$ is negligible.*

The following theorem shows that the non-adaptive OMSDH assumption is equivalent to the SDH assumption.

Theorem 4.4.1. *For any* $\mathsf{GroupGen}$*, any $n, d \in \mathbb{N}$ and a bound L, for any $\mathcal{Q}_x \subset \mathbb{F}$ of cardinality* $\mathsf{poly}(\lambda)$ *and for any PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{B}$ such that:*

$$\mathbf{Adv}^{(n,d)\text{-}\mathtt{OMSDH}}_{\mathsf{GroupGen},\mathcal{A}}(\lambda) = \mathbf{Adv}^{(n^2+d+1)\text{-}\mathtt{SDH}}_{\mathsf{GroupGen},\mathcal{B}}(\lambda).$$

Moreover, let $\mathcal{Q}_{\mathcal{A}}$ the query made by an adversary $\mathcal{A}$. For any $m \in \mathbb{N}$, for any PPT $\mathcal{A}$ such that $\max\{i : (x, i) \in \mathcal{Q}_{\mathcal{A}}\} \leq m$*:*

$$\mathbf{Adv}^{(n,d)\text{-}\mathtt{OMSDH}}_{\mathsf{GroupGen},\mathcal{A}}(\lambda) = \mathbf{Adv}^{(nm+d+1)\text{-}\mathtt{SDH}}_{\mathsf{GroupGen},\mathcal{B}}(\lambda).$$

In the proof, we define a reduction to the SDH assumption and whose simulation strategy works, similarly to [FFK+23, TZ23], only for queries on the points $\mathcal{Q}_x$.

Proof. Let m be (an upper bound to) the maximum power i queried to $\mathcal{O}_s$ and let $q' := nm + d + 1$, notice that the worst case is $m = n$. We show a reduction $\mathcal{B}$ to the q'-SDH assumption.

$\mathcal{B}$ takes as input $\mathsf{srs}' \leftarrow (\left[(s^i)_{i\in[0,q']}\right]_1, [1, s]_2)$ and defines the new SRS:

$$\mathsf{srs} \leftarrow \left(\left[(p(s)s^i)_{i\in[0,d]}\right]_1, [1, s]_2\right)$$

where $p(X) := (X - x_r)\prod_{x\in\mathcal{Q}_x}(X - x)^n$ and x_r is a random point. In particular, the group generator $[1]_1$ given to the adversary is randomized (and equal to $[p(s)]_1$ in the basis of the reduction).

The reduction can easily answer, for any $x_j \in \mathcal{Q}_x$ and any $i \leq m$, when the adversary queries the oracle $\mathcal{O}_s$ with (x_j, i): it just computes $z \leftarrow [p(s)(s-x_j)^{-i}]_1$, that is a valid output since $e(z, [s-x_j]_2^i) = [1]_T$.

Finally, notice that the forgery for $\mathcal{A}$ is $[p(s)/(s-x^*)]_1$. If we compute the Euclidean division between polynomials we obtain $q(X)$ and r such that:

$$\frac{p(X)}{(X-x^*)} = q(X) + \frac{r}{X-x^*}.$$

Noticing that $x^* \notin \mathcal{Q}_x$ implies, by construction of $p(X)$, that x^* does not divide $p(X)$, then $r \neq 0$. Therefore, the forgery of the reduction $\mathcal{B}$ is set to $(\mathsf{y} - [q(s)]_1)r^{-1}$. □

Hereafter, we show that the OMSDH assumption holds in the generic bilinear Maurer's version of the GGM [Mau05, ZZ21], in which an adversary can access elements from the groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ only via abstract handles. These are maintained by the challenger in lists L_1, L_2, and L_T, which correspond to the three groups. Similarly to [BFHK23], we consider the variant proposed by Zhandry [Zha22], where the adversary cannot choose the handle where the result is stored nor access handles not explicitly given as an oracle reply.

One-more d-SDH in the Maurer's GGM. The list L_1 initially contains the handles to the elements $1, Z, \ldots, Z^d$, and the list L_2 contains the handle to the elements $1, Z$. The challenger samples $z \leftarrow\!\!\$\ \mathbb{F}_q$ as the solution: the goal is to prove that z remains information-theoretically hidden from the adversary. Since the OMSDH assumption is interactive, we also give the adversary access to the oracle $\mathcal{O}_s$, and the handles returned by this oracle are stored in the list L_s.

The adversary is granted access to three types of oracles:

Group Oracles: for $i \in \{1, 2, T\}$, the oracle $\mathcal{O}_i$ takes as input two handles $h_1, h_2 \in L_i$ for polynomials $P_1(Z)$, $P_2(Z)$ and outputs a handle h for the polynomial $P_1(Z) + P_2(Z)$; L_i is accordingly updated with the handle h.

Paring Oracle: the oracle $\mathcal{O}_e$ on input two handles $h_1 \in L_1$ and $h_2 \in L_2$ for $P_1(Z)$ and $P_2(Z)$, outputs the handle h_T to the element $P_1(Z) \cdot P_2(Z)$ and updates L_T accordingly.

SDH Oracle: The oracle $\mathcal{O}_s$ takes as input a pair (x, i) and returns and handle h_s for $(Z-x)^{-i}$, updating L_s accordingly.

Following the standard argument in the GGM, we need to bound the probability that the so-called *collision event* E occurs, namely that there exist two handles h_1, h_2 that point to two distinct polynomials $P_1(Z)$ and $P_2(Z)$ and such that $P_1(Z) \neq P_2(Z)$, but $P_1(z) = P_2(z)$.

Definition 4.4.2. *We say that the n-one-more d-SDH assumption is secure in the Maurer's bilinear GGM if for any no-uniform PT adversary $\mathcal{A}$ with oracle interfaces described above triggers the collision event with almost negligible probability in λ.*

Theorem 4.4.2. *The n-one-more d-SDH assumption is secure in the Maurer's bilinear GGM.*

Proof. We notice that any no-uniform PT adversary for the OMSDH assumption in the GGM is inherently non-adaptive. In fact, if we run twice $\mathcal{A}$ then it would output the same queries to its SDH oracle, unless in one of the executions the collision event happens, in that case, the

$\mathbf{Exp}^{\Phi\text{-}se}_{\mathcal{A},\mathcal{S},\mathcal{E}}(\lambda)$

$\mathsf{pp}_{\mathbb{G}} \leftarrow\!\!\$\ \mathsf{GroupGen}(1^\lambda)$
$\mathsf{pp}_\Phi \leftarrow\!\!\$\ \Phi_0(\mathsf{pp}_{\mathbb{G}})$
$(\mathsf{srs}, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}(0, \mathsf{pp}_{\mathbb{G}})$
$(\mathbb{x}, \pi, \mathsf{aux}_{\mathcal{E}}, \mathsf{aux}_\Phi) \leftarrow \mathcal{A}^{\mathcal{S}_1, \mathcal{S}_2}(\mathsf{srs}, \mathsf{pp}_\Phi)$
$\mathbb{w}_{\mathcal{F}} \leftarrow \mathcal{E}(\mathsf{srs}, \mathbb{x}, \pi, \mathsf{aux}_{\mathcal{E}})$
$\mathsf{view} \leftarrow (\mathsf{srs}, \mathsf{pp}_\Phi, \mathcal{Q}_{\mathsf{sim}}, \mathcal{Q}_{\mathsf{RO}}, \mathcal{Q}_{\mathsf{aux}})$
$b_\Phi \leftarrow \Phi_1((\mathbb{x}, \pi), \mathsf{view}, \mathsf{aux}_\Phi)$
$b_V \leftarrow \mathsf{Verify}^{\mathcal{S}_2}(\mathsf{srs}, \mathbb{x}, \pi)$
$b_{\mathcal{E}} \leftarrow \forall\, \mathbb{w} : \mathcal{F}(\mathbb{w}) \neq \mathbb{w}_{\mathcal{F}} \vee (\mathsf{pp}, \mathbb{x}, \mathbb{w}) \notin \mathcal{R}$
return $(b_\Phi \wedge b_V \wedge b_{\mathcal{E}})$

$\mathcal{S}_1(\mathbb{x}, \mathsf{aux})$

$\pi, \mathsf{st}_{\mathcal{S}} \leftarrow \mathcal{S}(1, \mathsf{st}_{\mathcal{S}}, \mathbb{x}, \mathsf{aux})$
$\mathcal{Q}_{\mathsf{sim}} \leftarrow \mathcal{Q}_{\mathsf{sim}} \cup \{(\mathbb{x}, \mathsf{aux}, \pi)\}$
return π

$\mathcal{S}_2(s, \mathsf{aux})$

if $\nexists \mathsf{aux}, a : (s, \mathsf{aux}, a) \in \mathcal{Q}_{\mathsf{RO}}$:
  $a, \mathsf{st}_{\mathcal{S}} \leftarrow \mathcal{S}(2, \mathsf{st}_{\mathcal{S}}, s, \mathsf{aux})$
  $\mathcal{Q}_{\mathsf{RO}} \leftarrow \mathcal{Q}_{\mathsf{RO}} \cup \{(s, \mathsf{aux}, a)\}$
return a

Figure 4.1: The $(\Phi, \mathcal{F})$-SE experiments in ROM.

set of queries made by $\mathcal{A}$ in one execution is a subset of the set of queries made by $\mathcal{A}$ in the other execution.

Finally, by the composition lemma of the AGM [FKL18, Lemma 2.2], by Theorem 4.4.1 and since the SDH assumption holds in the Maurer's GGM, the theorem follows. □

4.5 Simulation-Extractable CP-SNARKS in the AGM

We extend the definitional framework of Section 3.4 to the $\mathcal{F}$-extractability setting, introduced by [BCKL08], where the extractor extracts a function of the witness. Notice that the simulation policy may depend on the function $\mathcal{F}$. Clearly, when $\mathcal{F}$ is the identity function, we obtain the policy-based notion of simulation extractability of Definition 3.4.4. We define the policy-based SE game in Fig. 4.1. In the figure, the extraction policy Φ takes as input the public view of the adversary view (namely, all the inputs received and all the queries and answers to its oracles). The set $\mathcal{Q}_{\mathsf{sim}}$ is the set of queries and answers to the simulation oracle. The set $\mathcal{Q}_{\mathsf{RO}}$ is the set of queries and answers to the random oracle. The set $\mathcal{Q}_{\mathsf{aux}}$ is the set of all the auxiliary information sent by the adversary (depending on the policy, this set might be empty or not). All these sets are initially empty and stored in the state of the simulator. The oracles $\mathcal{S}_1$ and $\mathcal{S}_2$ deal respectively with the simulation queries and the random oracle queries of $\mathcal{A}$.

Definition 4.5.1 (Φ-Simulation $\mathcal{F}$-extractability)**.** *A NIZK Π for a relation $\mathcal{R}$ and simulator $\mathcal{S}$ is $(\Phi, \mathcal{F})$-simulation extractable in the SRS model if for every PPT adversary $\mathcal{A}$ there exists an efficient extractor $\mathcal{E}$ such that the following advantage is negligible in λ:*

$$\mathbf{Adv}^{\Phi\text{-}se}_{\Pi,\mathcal{A},\mathcal{S},\mathcal{E}}(\lambda) \coloneqq \Pr\left[\mathbf{Exp}^{\Phi\text{-}se}_{\Pi,\mathcal{A},\mathcal{S},\mathcal{E}}(\lambda) = 1\right]$$

Moreover, given a family of policies $\boldsymbol{\Phi}$ and a family of functions $\boldsymbol{\mathcal{F}}$, we say that a NIZK Π is $(\boldsymbol{\Phi}, \boldsymbol{\mathcal{F}})$-simulation-extractable if Π is $(\Phi, \mathcal{F})$-simulation-extractable for any $\Phi \in \boldsymbol{\Phi}$ and $\mathcal{F} \in \boldsymbol{\mathcal{F}}$. We say that Π is $\boldsymbol{\Phi}$-simulation-extractable if Π is $(\boldsymbol{\Phi}, id)$-simulation-extractable and id is the identity function.

4.5.1 Simulation Extractability for KZG-based CP-SNARKs

We specialize the notion introduced in the previous section for CP-SNARKs based on the KZG commitment scheme. First, we specialize the definition of policy-based SE to the algebraic group model.

Definition 4.5.2 (Simulation extractability in the AGM)**.** *Let* Π *be a NIZK for a relation* $\mathcal{R}$ *with a simulator* $\mathcal{S}$*.* Π *is* $(\Phi,\mathcal{F})$*-simulation-extractable (or simply* $(\Phi,\mathcal{F})$*-SE) if there exists an efficient extractor* $\mathcal{E}$ *such that for every PPT algebraic adversary* $\mathcal{A}$*, the advantage* $\mathbf{Adv}^{\Phi\text{-}se}_{\Pi,\mathcal{A},\mathcal{S},\mathcal{E}}(\lambda) \in \mathsf{negl}(\lambda)$*.*

KZG-based CP-SNARKs. We recall that the KZG commitment scheme allows for simple and efficient *evaluation proofs* which, in the framework of [CFQ19], is a CP-SNARK for the relation $\mathcal{R}_{\mathtt{evl}}((\mathsf{c},x,y),f) = 1$ if and only if $f(x) = y \wedge \mathsf{c} = [f(s)]_1$. Informally, we say that a CP-SNARK is KZG-based if it internally calls, implicitly or explicitly, the CP-SNARK $\mathsf{CP}_{\mathtt{evl}}$ defined in the previous paragraph. This definition is rather informal, thus, we give below a formal notion that includes all the KZG-based CP-SNARKs.

Definition 4.5.3 (KZG-based CP-SNARK)**.** *We say that* CP *is KZG-based if* CP *is a CP-SNARK (for some relation* $\mathcal{R}$ *and) for the KZG commitment scheme, where: the proofs can be parsed as vectors of elements in* $\mathbb{G}_1$ *and* $\mathbb{F}$*, and the verification on* $(\mathbb{x},\pi)$ *consists of equations of the form:*

$$\textstyle\sum_i e(\mathsf{x}_i,[p_i(s)]_2) + \sum_i e(\mathsf{q}_i,[p'_i(s)]_2) = [p''(s)]_T$$

where $(\mathsf{x}_i)_i$ *are the* $\mathbb{G}_1$*-elements of the instance* $\mathbb{x}$*,* $(\mathsf{q}_i)_i$ *are the* $\mathbb{G}_1$*-elements of the proof* π*, and the (linear) polynomials* p_i, p'_i *and* p'' *are functions of the instance* $\mathbb{x}$*, the proof* π*, and possibly of the random oracle.*

Algebraic Consistency. In Definition 3.5.1 we defined a necessary property to achieve extractability in the presence of a simulation oracle for any KZG-based SNARKs. The property is motivated by the generalization of the simple attack described in Remark 3.5.1 and that we briefly recall. If for a commitment c an adversary is given two simulated KZG evaluation proofs π_1,π_2 on the same evaluation point x and for two different evaluation values y_1 and y_2, by the homomorphic property of KZG, the adversary can forge an evaluation proof on the statement $((\alpha+\beta)\mathsf{c}, x, \alpha y_1+\beta y_2)$ by setting the proof $\alpha\pi_1+\beta\pi_2$. This attack can be generalized whenever the adversary can leverage *algebraic inconsistencies* provided by simulated proofs, as we explain hereafter.

Let $\vec{A} \in \mathbb{F}[X]^{m\times n}$, and let $\vec{b} \in \mathbb{F}[X]^m$. We have that $(\vec{A}\|\vec{b})$ describes a linear system of polynomial equations that admits a solution if there exists a vector $\vec{z} \in (\mathbb{F}[X])^n$ such that $\vec{A}\cdot\vec{z} = \vec{b}$.

Definition 4.5.4 (Algebraic Consistency)**.** *Let* Π *be a KZG-based CP-SNARK. Let* view *be the view of* $\mathcal{A}$ *at the end of the SE game* $\mathbf{Exp}^{\Phi\text{-}se}_{\Pi,\mathcal{A},\mathcal{S},\mathcal{E}}$ *for an adversary* $\mathcal{A}$*. We say that the view* view *is* algebraic consistent *if the linear system* S *of polynomial equations, that we describe next, admits a solution.*

Let coms *be the list of simulated commitments in* pp_Φ*, where* $\mathsf{coms} \coloneqq (\mathsf{c}_k)_k$ *and* $\forall k : \mathsf{c}_k \in \mathbb{G}_1$*, and* proofs *be the list of simulated proofs* $\mathsf{proofs} \coloneqq (\pi_k)_k$ *(where* $\pi_k \coloneqq (\mathsf{q}_{k,j})_j, \vec{y}_k$ *and* $\forall k,j : \mathsf{q}_{k,j} \in \mathbb{G}_1, y_{k,j} \in \mathbb{F}$*) included in the view* view*. We assign to each simulated commitment* c_k *in* view

a formal variable (defining a polynomial) Z_k, similarly we assign to each $\mathbb{G}_1$-group element of the simulated proofs $\mathsf{q}_{k,j}$ formal variables (defining polynomials) $Q_{k,j} \in \mathbb{F}_{\leq d}[X]$. For each simulation query we define new equations derived by the verification equations of Π and from the algebraic representations of the instances queried to the simulation oracle. In particular, for the k-th simulation query with instance $\mathbb{x}_k$ and whose $\mathbb{G}_1$-elements are $(\mathsf{x}_{k,j})_j$ and simulated proof π_k, we can associate the polynomials of the verification equation $p_{k,i}, p'_{k,i}$ and p''_k, and we add the following equation to the linear system of polynomial equations S:

$$\sum_i \left(f_{k,i}(X) + \langle \vec{c}_{k,i}, \vec{Z}\rangle + \langle \vec{o}_{k,i}, \vec{Q}\rangle\right) p_{k,i}(X) + \sum_i Q_{k,i} \cdot p'_{k,i}(X) = p''_k(X)$$

where $\vec{Z}$ is the vector of all variables Z_j for any j and $\vec{Q}$ is the vector of all the variables $Q_{i,j}$ for any i, j, and the algebraic representation of $\mathsf{x}_{k,i}$ is $(\vec{f}_{k,i}, \vec{c}_{k,i}, \vec{o}_{k,i})$ and $f_{k,i}(X) = \sum_j (\vec{f}_{k,i})_j X^j$.

As a concrete example, for the KZG-based CP-SNARK $\mathsf{CP}_{\mathtt{evl}}$, from the k-th simulation query with instance (c, x, y) we can derive and add to the linear system of polynomial equations the equation:

$$(f(X) + \langle \vec{c}, \vec{Z}\rangle + \langle \vec{o}, \vec{Q}\rangle) - y - Q_k(X - x) = 0$$

where $\mathsf{c} = [f(s)]_1 + \langle \vec{c}, \mathsf{coms}\rangle + \langle \vec{o}, \mathsf{proofs}\rangle$. Notice that once we have computed a solution for S, the linear system of polynomial equations, we can represent it in a reduced form.

Definition 4.5.5 (Reduced solution)**.** *Given a solution $(\vec{z}, \vec{q})$ for a linear system S defining the algebraic consistency of SE experiment (see Definition 4.5.4), we say that $\vec{z}$ is its* reduced solution.

Given a *reduced solution* for S, it is possible to determine the (non-rational) polynomials $q_i(X)$. In fact, once we assign the values for the variables $\vec{Z}$ to $\vec{z}$, the linear system has $|\mathsf{proofs}|$ variables and $|\mathsf{proofs}|$ (independent) equations, thus admits one solution.

4.5.1.1 Strengthening the simulation extractability of KZG

In Chapter 3, KZG was proved simulation-extractable under a semi-adaptive policy. The main limits of that policy are that the adversary can query simulated proofs on instances (c_j, x_j, y_j) where only the evaluation values y_j can be adaptively chosen. Instead, the evaluation points x_j must be selectively chosen independently of the commitment key, and the commitments c_j cannot depend on the simulated proofs. In this chapter we prove that KZG achieves simulation extractability in AGM and RO:

1. under q-SDH assumption, for a semi-adaptive policy more flexible than the one in Chapter 3
2. under the OMSDH assumption, for a fully adaptive policy.

We describe our extraction policy. First, we notice that to prove simulation extractability for the KZG-based CP-SNARK $\mathsf{CP}_{\mathtt{m\text{-}evl}}$ (and in general for any KZG-based CP-SNARK), we can consider the (stronger) SE experiment where the simulation oracle returns simulated proofs for $\mathsf{CP}_{\mathtt{evl}}$. In fact, we can consider the reduction that, at any simulation oracle call for $\mathsf{CP}_{\mathtt{m\text{-}evl}}$ from the adversary, would first call the simulation oracle for $\mathsf{CP}_{\mathtt{evl}}$ and then assemble a valid simulated proof for $\mathsf{CP}_{\mathtt{m\text{-}evl}}$.

To enable the adversary to ask simulation proofs for commitments c whose representation depends on previously obtained simulated proofs (what we call a *proof of a proof*), we need to introduce the following definition.

Definition 4.5.6 (Nesting level of a proof)**.** *Let* view *be the view of an adversary at the end of the SE game for a KZG-based CP-SNARK, and let* $x_1, \ldots, x_n$ *be the list of all the evaluation points in the simulation queries. For each (single-eval) simulation statement* $((\mathsf{c}, x, y), \pi) \in \mathcal{Q}_{\mathsf{sim}}$ *let* $\vec{c}$ *(resp.* $\vec{o}$*) be the coefficients associated with the commitments* $\mathsf{coms} := (\mathsf{c}_j)_j$ *(resp. simulated proofs* $\mathsf{proofs} := (\pi_j)_j$*) in the algebraic representation of* c*. Let* b_k *be equal to* 1 *if* $x = x_k$ *and* 0 *otherwise. Let* $b_{j,k}$ *be equal to* 1 *if* $x = x_k$ *and* $\mathsf{c}_j \neq 0$*, and* 0 *otherwise.*

For all $j \in [|\mathsf{coms}|]$*,* $k \in [n]$*, the nesting level* $\nu_\pi(j,k)$ *of the simulated proof* π *on the simulated commitment* c_j *and the point* x_k *is equal to:*

$$\nu_\pi(j,k) := \max_{i : o_i \neq 0 \wedge \nu_{\pi_i}(j,k) \neq 0} \{\nu_{\pi_i}(j,k) + b_k\} \cup \{b_{j,k}\}$$

We define the maximum nesting level $\bar{\nu} := \max_j \sum_k \max_{\pi_i} \nu_{\pi_i}(j,k)$.

Informally, the idea behind the maximum nesting level $\bar{\nu}$ is that each *proof of a proof* involving at some point one of the simulated commitments can (possibly but not always) increase the degree of the denominator of the rational function associated with such a simulated proof. The value $\bar{\nu}$ is the minimal upper bound on the degree of (the denominators of) the rational functions associated with the simulated proofs (see Lemma 4.3.1). We consider the following constraints, parametrized by a set $\mathcal{I} \subseteq [n]$.

Point check: given a set of points $\mathcal{Q}_x \in \mathsf{pp}_\Phi$, return 1 if $\forall \mathbb{x}$ queried to $\mathcal{S}_1$, we have that $\mathbb{x}.x \in \mathcal{Q}_x$

Hash check with Linearized Commitment (and parameter $\mathcal{I}$): Parsing the forgery instance $\mathbb{x}^* := (x^*, (\mathsf{c}_i^*, y_i^*)_{i \in [n]})$, return 1 if and only if there exist group elements $(\mathsf{b}_{i,r})_r$, polynomials $A_{i,r}(X)$, a non-constant polynomial h such that:

- $\forall i \in [n] : \mathsf{c}_i^* = \sum_r A_{i,r}(x^*)\mathsf{b}_{i,r}$
- $\forall i \in \mathcal{I} : ((\mathsf{b}_{i,r})_r; (A_{i,r})_r, h) \to_{\mathsf{RO}} a.$
- $h(a) = x^*$
- $\forall i \in \mathcal{I} : \{A_{i,r}\}_r$ are $\bar{\nu}$-independent polynomials, where $\bar{\nu}$ is the maximum nesting level (cf. Definition 4.5.6)

Looking ahead, the *point check* does require some form of programmability of the RO at SNARK level, while the *hash check* (with linearized commitment) essentially consists of checking the hash of the "virtual" representation of a group element and is weak enough to capture schemes tailored for optimizations, like the linearization trick [GWC19, OL].

Definition 4.5.7. *Let* $\mathbf{\Phi}^{\mathtt{adpt}}_{\mathtt{m\text{-}evl},\mathcal{I}}$ *(resp.* $\mathbf{\Phi}^{\mathtt{s\text{-}adpt}}_{\mathtt{m\text{-}evl},\mathcal{I}}$*) be the set of policies* $\Phi_{\mathcal{D}} = (\Phi_0^{\mathcal{D}}, \Phi_1)$ *for a distribution* $\mathcal{D}$ *where:*

- $\Phi_0^{\mathcal{D}}$ *on input group parameters* $\mathsf{pp}_{\mathbb{G}}$ *outputs* $\mathsf{pp}_\Phi := \mathsf{coms}$*, where* coms *is a vector of commitments sampled from* $\mathcal{D}$ *(resp. additionally it outputs a set* $\mathcal{Q}_x \subseteq \mathbb{F}$*).*

	[FFK+23]	$\mathbf{\Phi}^{\text{s-adpt}}_{\text{m-evl}}$	$\mathbf{\Phi}^{\text{adpt}}_{\text{m-evl}}$
Hash check w/ L.C.		✓	✓
Hash check	✓		
Point check	✓	✓	
Commitment check	✓		
Assumption (AGM)	$(Q+d+1)$-DL	$(Q+d+1)$-DL	(Q,d)-OMSDH

Table 4.1: Comparison of extraction policies in terms of constraints and security assumptions with related work.

- Φ_1 *is the hash check with parameter* $\mathcal{I}$ *defined above (resp.* Φ_1 *is the logical conjunction of the hash check, with parameter* $\mathcal{I}$*, and the point check).*
- $\mathcal{D}$ *is witness sampleable and the* $\mathcal{D}$*-Aff-MDH assumption holds.*

In Table 4.1 we compare our new extraction policies with the extraction policy of [FFK+23]. We stress that our hash check with linearized commitment is more permissive than their hash check constraint, therefore, our theorem is stronger. In the table, Q is the number of simulation queries and d is the maximum degree supported by the scheme.

For any set $\mathcal{I} \subseteq [n]$, let us denote with $\sigma_{\mathcal{I}}$ the $\mathcal{I}$-projection function, namely the function that takes as input a list $(a_1, \ldots, a_n)$ and returns the list $(a_i)_{i\in\mathcal{I}}$.

Theorem 4.5.1. $\forall \mathcal{I} \subseteq [n]$, $\mathsf{CP}_{\text{m-evl}}$ *is* $(\mathbf{\Phi}^{\text{adpt}}_{\text{m-evl},\mathcal{I}}, \sigma_{\mathcal{I}})$*-SE under the OMSDH assumption and is* $(\mathbf{\Phi}^{\text{s-adpt}}_{\text{m-evl},\mathcal{I}}, \sigma_{\mathcal{I}})$*-SE under the* $(Q_{\mathsf{sim}} + d)$*-dlog assumption in the AGM.*

Proof intuition. We consider an algebraic adversary $\mathcal{A}$ whose forgery satisfies the extraction policy. In particular, the view is algebraic consistent, thus there exists a solution for the polynomial system of linear equations defined by the view. As the first important step of the proof, we simplify this system of equations and find alternative representations where each simulated proof depends either on one single simulated commitment or on one single simulated proof. This simplification allows rewriting the forged linearized commitment in the more manageable form $\mathsf{c}^* = [m_0(s)]_1 + \sum [\log(\mathsf{c}_i) \cdot m_i(s)]_1$ where c_i are the simulated commitments. Here, we can prove that $m_i(X) \equiv 0$ for $i > 0$. In fact, assume otherwise and assume the commitments are uniformly random[9], then we can break the representation problem finding $\log(\sum m_i(x^*)\mathsf{c}_i) = y^* - m_0(x^*)$ where the forgery of the adversary is (c^*, x^*, y^*).

We are still not done because $m_0(X)$ is a rational function of the form

$$f(X) - \sum A_i(X)(\sum_j o_j q_{i,j}(X))$$

where f is the polynomial we would like to extract, the $q_{i,j}$ are rational functions whose degree is bounded by the maximum nesting level $\bar{\nu}$ and the o_j are the coefficients in the algebraic representation of c^* that depend on the simulated proofs. If we assume that the forgery is valid then we would obtain $m_0(x^*) = y^*$, otherwise we could break the OMSDH assumption, moreover, we can show this case happens when $\sum A_i(x^*)(\sum_j o_j q_i(x^*)) = 0$ but, the extractor

[9]In our proof we consider the more general case where the simulated commitments are sampled from an Aff-MDH-secure distribution.

would still fail if there exists at least one $o_j \neq 0$. Here we crucially use our hypothesis on $\bar{\nu}$-independence of the A_i to show that this cannot happen and thus all $o_j = 0$.

One might wonder if this last step is an artifact of our proof technique, and whether the independence is necessary. We show the latter is the case with an attack similar to the one presented in Section 4.2.3. The attack asks for a simulated proof on $(\mathsf{c}, 0, 1)$ for a simulated commitment c and sets the forged linearized commitment to $\mathsf{c}^* = \mathsf{c} - x^*\pi$ for an arbitrary evaluation point x^* and $y^* = 1$, the attack works because $\mathsf{c}^* - [1]_1 = \mathsf{c} - x^*\mathsf{c}/s - x^*/s + 1 = \pi(s - x^*)$. The formal polynomial associated to c^* would be of the form $0 - (1 - X \cdot \frac{1}{X}) + Z(1 - X \cdot \frac{1}{X})$ where Z is the formal variable associated to the simulated commitment, $o_1 = 1$ and $A_1(X) = 1$ and $A_2(X) = -X$ and where the latter polynomials are 1-linearly dependent.

Proof of Theorem 4.5.1. We recall that the set $\mathcal{I}$ contains the indexes i such that $\mathcal{E}$ needs to extract the witness polynomials f_i committed in c_i^*.

By the definition of algebraic adversary (cf. Definition 2.2.2) for each group element output, $\mathcal{A}$ additionally attaches a representation $(f, \vec{r})$ of such a group element with respect to all the elements seen during the experiment (included elements in coms and the simulated proofs). In particular, we assume that for each query $(\mathbb{x}, \mathsf{aux})$ to the oracle $\mathcal{S}_1$ we can parse the value aux as $((f_i, \vec{r}_i)_i, \mathsf{aux}')$, where $(f_i, \vec{r}_i)$ is a valid representation for $\mathbb{x}.\mathsf{c}_i$.

The adversary also encodes a polynomial $h(X)$ in aux_ϕ. The commitments c_i^* of the forgery come along with representation $(A_{i,r}(X), \mathsf{b}_{i,r})_r$, stored in $\mathsf{aux}_\mathcal{E}$; the adversary also stores $(f_{\mathsf{b}_{i,r}}, \vec{r}_{\mathsf{b}_{i,r}})$ to represent the group element $\mathsf{b}_{i,r}$. Namely, given the commitments coms and all the proofs $\mathsf{proofs}_\mathcal{A}$ output by $\mathcal{S}_1$, it holds that $\mathsf{c}_i^* = \sum_r A_{i,r}(x^*)(f_{\mathsf{b}_{i,r}}(s) + \langle \vec{r}_{\mathsf{b}_{i,r}}, \mathsf{coms} \| \mathsf{proofs}_\mathcal{A} \rangle)$.

Without loss of generality, we restrict the class of the algebraic adversaries that we consider. Given an algebraic adversary $\mathcal{A}$ we can define a new adversary $\mathcal{A}'$ such that:

- $\mathcal{A}'$ makes (single-eval) simulation queries, i.e., each statement $\mathbb{x}$ given as input to $\mathcal{S}_1$ can be parsed as (c, x, y)
- each commitment in $\mathbb{x}$ is a linear combination of simulated commitments and proofs, but not of elements of the SRS

The adversary $\mathcal{A}'$ runs internally $\mathcal{A}$ and forwards all its queries and answers to the simulation oracle in the following way:

- Upon query $\mathbb{x} := (x, (\mathsf{c}_i, y_i))$ to $\mathcal{S}_1$, with representation $(f_i, \vec{r}_i)$ such that $\mathsf{c}_i = [f_i(s)]_1 + \langle \vec{r}_i, \mathsf{coms} \| \mathsf{proofs}_\mathcal{A} \rangle$, $\mathcal{A}'$ queries n times the simulation oracle with $(\mathsf{c}_i - [f_i(s)]_1, x, y_i - f_i(x))$, receiving the proof π_i. Outputs the proof $\pi' := \sum_i \rho^{i-1}(\pi_i + \mathsf{Prove}_{\mathtt{evl}}(\mathsf{ek}, ([f_i(s)]_1, x, f_i(x)), f))$, where $\rho \leftarrow \mathsf{RO}(\mathsf{batch} \| \mathbb{x})$

By the homomorphic properties of $\mathsf{CP}_{\mathtt{m\text{-}evl}}$, the correctness of the proofs readily holds.

We define our extractor $\mathcal{E}$ to be the extractor that returns, for all $i \in \mathcal{I}$ the polynomial $f_i(X) := \sum_r A_{i,r}(x^*) f_{\mathsf{b}_{i,r}}(X)$; this turns out to be equivalent to the canonical extractor in the AGM, because, as we show, the remaining entries in the representation sum up to zero.

We let $\mathbf{H}_0$ be the $\mathbf{Exp}^{\Phi\text{-}se}_{\mathcal{A},\mathcal{S},\mathcal{E}}$ experiment, and we denote by $\epsilon_i := \Pr[\mathbf{H}_i = 1]$.

Hybrid $\mathbf{H}_1$. We set $\mathbf{H}_1$ to be the same experiment but with the *alternative* adversary $\mathcal{A}'$ defined below:

1. The alternative adversary runs $\mathcal{A}$ forwarding all its queries until $\mathcal{A}$ sends its forgery. Let $\bar{\mathbb{x}} = (\bar{x}, (\bar{\mathsf{c}}_i, \bar{y}_i)_i), \bar{\pi}$ be its forgery. Let $\mathcal{Q}_x$ be the set of points x_j for which the adversary queried $\mathcal{S}_1$.
2. If $\bar{x} \in \mathcal{Q}_x$, namely when the adversary made a simulation query with evaluation point set to $\bar{x}$, then it finds values y_i such that $\mathbb{x} \coloneqq (\bar{x}, (\bar{\mathsf{c}}_i, y_i)_i)$ is algebraic consistent with the view of the adversary, and queries the simulation oracle $\mathcal{S}_1$ with $\mathbb{x}'$ receiving back π. (Else it outputs $\bar{\mathbb{x}}, \bar{\pi}$.)
3. It computes the forgery $\mathbb{x}^* = (\mathsf{c}^*, x^*, y^*), \pi^*$, where:

$$\mathsf{c}^* \leftarrow (\bar{\pi} - \pi) \qquad \pi^* \leftarrow \frac{\bar{\pi} - \pi}{\bar{x} - x^*} \qquad y^* \leftarrow \frac{\sum_i \rho^{i-1}(\bar{y} - y)}{\bar{x} - x^*}$$

   the forgery point $x^* \leftarrow \mathsf{RO}(s)$, and s is a string never queried to the RO by $\mathcal{A}$ and containing c^* as substring, which yields $\mathsf{c}^* \rightarrow_{\mathsf{RO}} x^*$.
4. It aborts if $x^* \in \mathcal{Q}_x$, otherwise it outputs the forgery.

We show that, unless it occurs the bad event that $x^* \in \mathcal{Q}_x$, the forgery of the adversary $\mathcal{A}'$ is valid whenever the forgery of $\mathcal{A}$ is valid. First we notice, by the verification equation of KZG, that $(\bar{\pi} - \pi)(s - \bar{x}) = \sum_i \rho^{i-1} [y_i - \bar{y}_i]_1$. Thus:

$$\pi^*(s - x^*) = \tfrac{\bar{\pi}-\pi}{\bar{x}-x^*}(s - x^* + \bar{x} - \bar{x}) = \tfrac{\bar{\pi}-\pi}{\bar{x}-x^*}(-x^* + \bar{x}) + \tfrac{\bar{\pi}-\pi}{x^*-x}(s - \bar{x}) = \mathsf{c}^* - [y^*]_1 \tag{4.1}$$

Moreover, the probability of the bad event is at most $\frac{Q_{\mathsf{RO}}+1}{q}$, where Q_{RO} is the number of queries of $\mathcal{A}$ to the random oracle. We have that $\epsilon_1 \leq \epsilon_0 + \frac{Q_{\mathsf{RO}}+1}{q}$

Hybrid $\mathbf{H}_2$. Recall that $\mathcal{D}$ is witness sampleable, thus according to Definition 2.3.3 there exists a PPT algorithm $\tilde{\mathcal{D}}$ associated with the sampler $\mathcal{D}$. The hybrid $\mathbf{H}_2$ is identical to the previous one, but the group elements in coms are sampled "at the exponent", i.e., we use $\tilde{\mathcal{D}}$ to generate the field elements $\vec{\gamma}$, and we let $\mathsf{coms} \leftarrow [\vec{\gamma}]_1$. By the witness sampleability of $\mathcal{D}$, $\mathbf{H}_1 \equiv \mathbf{H}_2$, thus $\epsilon_2 = \epsilon_1$.

Hybrid $\mathbf{H}_3$. In this hybrid we add some more entries to the list of simulated proofs $\mathcal{Q}_{\mathsf{sim}}$.

The experiment runs $\mathcal{A}$ until completion. Since the view of the adversary is algebraic consistent, we can define a set of polynomial equations that admits solutions derived from the simulation queries of $\mathcal{A}$. Let $(z_i(X))_{i \in [Q_{\mathsf{sim}}]}$ be a reduced solution (Definition 4.5.5) for this set of polynomial equations. The experiment submits additional queries to $\mathcal{S}_1$ as follows. First, for all j, k, define the value $\bar{\nu}_{j,k} \coloneqq \max_\pi \nu_\pi(j, k)$.

Then $\forall j, k$ such that $\bar{\nu}_{j,k} \neq 0$, does the following. It queries $\mathcal{S}_1$ with $(\mathsf{coms}_j, x_k, z_j(x_k))$, and let $\bar{\pi}_{j,k,1}$ be the output of $\mathcal{S}_1$ on each of these queries; then, for $l \in [\bar{\nu}_{j,k} - 1]$, obtains the proof $\bar{\pi}_{j,k,l+1}$ on the statement $(\bar{\pi}_{j,k,l}, x_k, q_{j,k,l}(x_k))$, where: $q_{j,k,1}(X) \coloneqq z_j(X)$ and $\forall l > 0$ the polynomial $q_{j,k,l+1}(X) \coloneqq (q_{j,k,l}(X) - q_{j,k,l}(x_k))(X - x_k)^{-1}$. We notice all these additional proofs are of the form:

$$\bar{\pi}_{j,k,l} = \left[\frac{\gamma_j - \sum_{l' \in [l]} (s - x_k)^{l'-1} q_{j,k,l'}(x_k)}{(s - x_k)^l} \right]_1 \tag{4.2}$$

We call them "core" proofs, as they will play an important role in the next hybrid, and we denote with proofs the vector of all the core proofs, to distinguish them from the adversary's proofs $\mathsf{proofs}_{\mathcal{A}}$, namely the set of simulated proofs requested by the adversary. These additional

simulation queries are algebraic consistent w.r.t. the view of $\mathcal{A}$: in particular, $\forall j, k$ each proof $\bar{\pi}_{j,k,1}$ is an evaluation proof for the commitment coms_j on the point x_k and the evaluation value $z_j(x_k)$, which by definition of the polynomials $z_j(X)$ is algebraic consistent. For all $l > 1$ the proof $\bar{\pi}_{j,k,l}$ is a proof on the point x_k for a commitment that is a quotient derived from $z_j(X)$, and the evaluation value is chosen to be consistent with it. The change introduced in this hybrid does not alter the winning probability of $\mathcal{A}$, hence $\epsilon_3 = \epsilon_2$.

Hybrid $\mathbf{H}_4$. In this hybrid we change the representation of the forgery of $\mathcal{A}$. In particular, once $\mathcal{A}$ has submitted the (successful) forgery $(\mathbb{x}^*, \pi^*)$, attaching its representation, we replace it with new coefficients that only depend on coms, and proofs, but not on the adversary's proofs $\mathsf{proofs}_{\mathcal{A}}$.

The change introduced in this hybrid is only syntactical and does not alter the winning probability of $\mathcal{A}$, as we show hereafter.

Lemma 4.5.1. $\epsilon_4 = \epsilon_3$

Proof. We give a recursive procedure that rewrites the algebraic representations of all the Q_{sim} adversary's proofs $\mathsf{proofs}_{\mathcal{A}}$ in the base defined by proofs. We prove by induction on the number of simulation queries made by the adversary the correctness of the procedure.
Base. Let π be the first proof computed by $\mathcal{S}_1$, for an instance (c, x_{k^*}, y), where the commitment $\mathsf{c} = \sum_j c_j \left[\gamma_j\right]_1$. We have that, by the correctness of the proof, $\pi(s - x_{k^*}) = \sum_j c_j \left[\gamma_j\right]_1 - y$. By algebraic consistency, we have that there exists an equation of the form $\sum_j (c_j Y_j(x_{k^*})) = y$ with variables (the coefficients of) the polynomials Y_j, and the list of polynomials $(z_j(X))_j$ is a reduced solution by the change introduced in $\mathbf{H}_3$. We derive that π is equal to:

$$\left[\frac{\sum_j c_j(\gamma_j - z_j(x_{k^*}))}{s - x_{k^*}}\right]_1 = \left[\frac{\sum_j c_j(\gamma_j - q_{j,k^*,1}(x_{k^*}))}{s - x_{k^*}}\right]_1 = \sum_j c_j \bar{\pi}_{j,k*,1}$$

Inductive Step. Let now assume that all the first t proofs can be expressed as linear combination of elements of proofs. We show that also the $(t+1)$-th proof can be written in the same way.

Let (c, x_{k^*}, y) be the $(t+1)$-th (valid) query submitted to $\mathcal{S}_1$, where $\mathsf{c} = \sum_j c_j \left[\gamma_j\right]_1 + \sum_{j \leq t} o_j \pi_j$ and $\mathsf{proofs}_{\mathcal{A}} = (\pi_j)_{j \in [Q_{\mathsf{sim}}]}$. By induction, there exist coefficients $o'_{j,k,l}$ such that: $\mathsf{c} = \sum_j c_j \left[\gamma_j\right]_1 + \sum_{j,k,l} o'_{j,k,l} \bar{\pi}_{j,k,l}$. The proof π computed by $\mathcal{S}_1$ (we set $\pi_{t+1} := \pi$) is such that $\pi(s - x_{k^*}) = \mathsf{c} - \left[y\right]_1$. Let $\pi = \left[p\right]_1$, then we have that:

$$p = \frac{1}{s - x_{k^*}} \left(\sum_j c_j \gamma_j + \sum_{j,k,l} o'_{j,k,l} \frac{\gamma_j - \sum_{l' \in [l]} (s - x_k)^{l'-1} q_{j,k,l'}(x_k)}{(s - x_k)^l} - y \right)$$

Also, $y = \sum_j c_j z_j(x_{k^*}) + \sum_{j,k,l} o'_{j,k,l} q_{j,k,l}(x_{k^*})$ by algebraic consistency, and it can be expanded as:

$$\sum_j c_j z_j(x_{k^*}) + \overbrace{\sum_{j,k \neq k^*,l} o'_{j,k,l} \frac{z_j(x_{k^*}) - \sum_{l' \in [l]} (x_{k^*} - x_k)^{l'-1} q_{j,k,l'}(x_k)}{(x_{k^*} - x_k)^l}}^{\bar{y}} + \sum_{j,l} o'_{j,k^*,l} q_{j,k^*,l}(x_{k^*})$$

We first notice that, by plugging the equation above, we can rewrite p as:

$$\sum_j c_j \tfrac{\gamma_j - z_j(x_{k^*})}{s-x_{k^*}} + r + \sum_{j,l} o'_{j,k^*,l} \tfrac{\gamma_j - \sum_{l' \in [l+1]} (s-x_{k^*})^{l'-1} q_{j,k^*,l'}(x_k^*)}{(s-x_{k^*})^{l+1}}$$

where $r = \frac{1}{s-x_{k^*}} \left(\sum_{j,k \neq k^*,l} o'_{j,k,l} \frac{\gamma_j - q_{j,k,l}(x_k)}{s-x_k} - \bar{y} \right)$. Note that the first and the third addends of $[p]_1$ are linear combination of elements in proofs. The only thing left to prove is that $[r]_1$ can be written as linear combination of elements of proofs.

Let $N_{j,k,l} \coloneqq \gamma_j - \sum_{l' \in [l]} (s-x_k)^{l'-1} q_{j,k,l'}(x_k)$. For all j,k,l, we have that:

$$\begin{aligned}
\frac{\bar{\pi}_{j,k,l}}{s-x_{k^*}} &= \frac{[N_{j,k,l}]_1}{(s-x_k)^l (s-x_{k^*})} \\
&= [N_{j,k,l}]_1 \left(\sum_{\ell=0}^{l-1} \underbrace{(-1)^\ell \frac{(x_k - x_{k^*})^{-(\ell+1)}}{(s-x_k)^{l-\ell}}}_{\alpha_{j,k,l,\ell}} + \underbrace{(-1)^l \frac{(x_k - x_{k^*})^{-l}}{(s-x_{k^*})}}_{\beta_{j,k,l}} \right)
\end{aligned}$$

Also, for all $\ell \in [0, l-1]$, we have that $[N_{j,k,l}]_1 \, \alpha_\ell$ is equal to:

$$(-1)^\ell \bar{\pi}_{j,k,l-\ell} + (-1)^{\ell+1} (x - x_{k^*})^{\ell-1} \sum_{l' \in [\ell]} [q_{j,k,l'}(x_k)]_1 \, (s-x_k)^{l'-1}$$

Also, we have that $[N_{j,k,l}]_1 \, \beta$ is equal to:

$$\begin{aligned}
(-1)^l \bar{\pi}_{j,k^*,l} + (-1)^{l+1} \left[\frac{z_j(x_{k^*}) - \sum_{l' \in [l]} (s-x_k)^{l'-1} q_{j,k,l'}(x_k)}{s-x_{k^*}} \right]_1 = \\
(-1)^l \bar{\pi}_{j,k^*,l} + \left[\frac{z_j(x_{k^*}) - \sum_{l' \in [l]} (x_k - x_{k^*})^{l'-1} q_{j,k,l'}(x_k)}{(s-x_{k^*})(x_k^* - x_k)^l} \right]_1 = \\
+ \sum_{\ell=0}^{l-1} (-1)^\ell (x - x_{k^*})^{\ell-1} \sum_{l' \in [\ell]} [q_{j,k,l'}(x_k)]_1 \, (s-x_k)^{l'-1}
\end{aligned}$$

Using the above equations, we can write $[r]_1$ as:

$$\begin{aligned}
&\sum_{j,k \neq k^*,l} o'_{j,k,l} \frac{[N_{j,k,l}]_1}{(s-x_k)^k (s-x_{k^*})} - \frac{[\bar{y}]_1}{s-x_{k^*}} = \\
&\sum_{j,k \neq k^*,l} o'_{j,k,l} \, [N_{j,k,l}]_1 \left(\sum_{\ell=0}^{l-1} \alpha_{j,k,l,\ell} + \beta_{j,k,l} \right) - \frac{[\bar{y}]_1}{s-x_{k^*}} = \\
&\sum_{j,k \neq k^*,l} o'_{j,k,l} \left(\sum_{\ell=0}^{l-1} (-1)^\ell \bar{\pi}_{j,k,l-\ell} + (-1)^l \bar{\pi}_{j,k^*,l} \right) + \frac{[\bar{y}]_1}{s-x_k^*} - \frac{[\bar{y}]_1}{s-x_{k^*}} = \\
&\sum_{j,k \neq k^*,l} o'_{j,k,l} \left(\sum_{\ell=0}^{l-1} (-1)^\ell \bar{\pi}_{j,k,l-\ell} + (-1)^l \bar{\pi}_{j,k^*,l} \right)
\end{aligned}$$

□

Before moving to the next hybrid, we set some notation. First, $\forall i, r$, let parse $\vec{r}_{\mathsf{b}_{i,r}} = \vec{c}_{i,r} \| \vec{o}_{i,r}$. From the definition of $\mathbf{H}_4$, we have that $\mathsf{c}_i^* = [f_i(s)]_1 + \sum_r A_{i,r}(x^*)(\langle \vec{c}_{i,r}, \mathsf{coms} \rangle + \langle \vec{o}_{i,r}, \mathsf{proofs} \rangle)$

From the change introduced in $\mathbf{H}_3$, $\mathcal{S}_1$ outputs "core" proofs $\bar{\pi}_{j,k,l}(s, \mathsf{coms}) \in \mathsf{proofs}$, where $\bar{\pi}_{j,k,l}(X, \vec{Y}) \coloneqq (Y_j - \sum_{l' \in [l]} (X - x_k)^{l'-1} q_{j,k,l'}(x_k))(X - x_k)^{-l}$.

By the guarantees of the AGM, for all $i \in [n]$ we can write $\mathsf{c}_i^* = [c_i^*(s, \mathsf{coms})]_1$, where $c_i^*(X, \vec{Y})$ is equal to:

$$\sum_r A_{i,r}(x^*) f_{\mathsf{b}_{i,r}}(X) + \sum_r A_{i,r}(x^*) \underbrace{\sum_j (c_{i,r,j} Y_j + \sum_{k,l} o_{i,r,j,k,l} \pi_{j,k,l}(X, \vec{Y}))}_{B_{i,r}(X,Y)} \tag{4.3}$$

and, if the verification equation is satisfied, we have that:

$$v(s) = \pi^*(s - x^*)$$

where $v(X) \coloneqq \sum_i \rho^{i-1}(c_i^*(X, \mathsf{coms}) - y_i^*)$.

Hybrid $\mathbf{H}_5$. This hybrid is equal to $\mathbf{H}_4$ but it returns 0 if there exists $i \in [n]$ such that $c_i^*(x^*, \mathsf{coms}) \neq y_i^*$.

Lemma 4.5.2. $\epsilon_5 \leq \epsilon_4 + \epsilon_{\mathsf{OMSDH}} + n/q$

Proof. If there exists $i \in [n]$ such that $c_i^*(x^*, \mathsf{coms}) \neq y_i^*$, with overwhelming probability $1 - n/q$, we have that $v(x^*) \neq 0$ because, by the hash check, ρ is chosen uniformly at random after the polynomials $c_i^*(X, \vec{Y})$ are determined. Then, we can make a forgery to the OMSDH[10] assumption as follows.

The reduction gives the adversary the same SRS generated by the OMSDH challenger. The oracle $\mathcal{O}_s$ allows the reduction to compute the proofs for any statement $\mathbb{x} \coloneqq (\mathsf{c}, x, y)$, where c is a linear combination of elements of the SRS, coms and previously seen simulated proofs. As shown in the previous hybrid, a proof for $\mathbb{x}$ can be computed using group elements of the form $[s - x_k]_1^{-l}$ (that can be retrieved using $\mathcal{O}_s$), the coefficients γ_j and the algebraic representation of c, which is all known to the reduction. Let $q(X), r$ be such that $v(X) = q(x) + r(X - x^*)$. The reduction submits the forgery (x^*, y^*), where $\mathsf{y}^* \coloneqq r^{-1}(\pi^* - [q(s)]_1)$. This is a valid forgery because y^* is equal to $[(s - x^*)^{-1}]_1$ and x^* was never queried to $\mathcal{O}$ by the change introduced in $\mathbf{H}_1$. □

Hybrid $\mathbf{H}_6$. Let $\mathbf{H}_6$ return 0 if there is an index $i \in \mathcal{I}$ such that f_i extracted by $\mathcal{E}$ is not a valid witness.

Lemma 4.5.3. $\epsilon_6 \leq \epsilon_5 + |\mathcal{I}| \epsilon_{\mathsf{Aff\text{-}MDH}} + \mathsf{deg}(h)(\sum_{i \in \mathcal{I}} \max_r \mathsf{deg}(A_{i,r}) + \bar{\nu})/q$

Proof. We prove it through a series of n hybrids. Let $\mathbf{H}_{6,0} \equiv \mathbf{H}_5$, and let $\mathbf{H}_{6,i}$ be the same as $\mathbf{H}_{6,i-1}$ and if $i \in \mathcal{I}$ it additionally returns 1 if f_i is not a valid witness. Clearly, for $i \notin \mathcal{I}$, it holds that $\epsilon_{6,i} = \epsilon_{6,i-1}$.

For $i \in \mathcal{I}$, let E_i be the event that $\sum_r A_{i,r}(x^*) B_{i,r}(X) \equiv 0$.

Case 1. We show that $\Pr[\mathbf{H}_{6,i} = 1 \wedge E_i] = 0$. We recall that the extractor $\mathcal{E}$ returns the polynomial $f_i(X) \coloneqq \sum_r A_{i,r}(x^*) f_{\mathsf{b}_{i,r}}(X)$. Conditioning on E_i, we have that $c_i^*(X, \vec{Y}) = f_i(X)$,

[10] When the policy is semi-adaptive, we can reduce to discrete log because of Theorem 4.4.1.

and $\mathsf{c}_i^* = [f_i(s)]_1$. $\mathcal{E}$ returns a valid witness if $f_i(x^*) = y_i^*$, which is enforced by the check introduced in $\mathbf{H}_5$.

Case 2. We show that $\Pr[\mathbf{H}_{6,i} = 1 \wedge \neg E_i] \leq \epsilon_{\mathsf{Aff\text{-}MDH}} + \mathsf{deg}(h)(\bar{\nu} + \max_r \mathsf{deg}(A_{i,r}))/q$. First, it must be that there exist indexes r^*, j^*, k^*, l^* such that either $c_{i,r^*,j^*} \neq 0$ or $o_{i,r^*,j^*,k^*,l^*} \neq 0$, as otherwise $B_{i,r} \equiv 0, \forall r$.

Let $\hat{c}_i(X, \vec{Y}) \coloneqq m_0(X) + \sum m_j(X) Y_j$ be a multilinear polynomial with coefficient in $\mathbb{F}_{\leq q}(X)$ where:

$$m_0(X) = f_i(X) - \sum_r A_{i,r}(X) \sum_{j,k,l} o_{i,r,j,k,l} \frac{\sum_{l' \in [l]} (X - x_k)^{l'-1} q_{j,k,l'}(x_k)}{(X - x_k)^l}$$

$$m_j(X) = \sum_r A_{i,r}(X) \underbrace{(c_{i,r,j} + \sum_{k,l} \frac{o_{i,r,j,k,l}}{(X - x_k)^l})}_{m_{j,r}(X)}, \ \forall j > 0$$

Notice that by definition we have that: $\hat{c}_i(x^*, \vec{Y}) = c_i^*(x^*, \vec{Y})$.

$\forall j$ let $p_j(X) \coloneqq \prod_k (X - x_k)^{\bar{\nu}_{j,k}}$, and notice that the set $\{p_j(x)\} \cup \{\frac{p_j(x)}{(X-x_k)^l}\}_{k,l}$ is an independent set of polynomials w.r.t $\mathbb{F}$ and $m_{j,k}(X) \cdot p_j(X)$ is in the span of such a set of polynomials, thus, because of the condition of Case 2, $m_{j^*,r^*}(X) \neq 0$.

Let $\nu_j^* \coloneqq \sum_k \bar{\nu}_{j,k}$. By definition, $\nu_j^* \leq \bar{\nu}$ that we recall is equal to $\max_j \sum_k \bar{\nu}_{j,k}$. Because of the *Hash check*, we have that $\{A_{i,r}\}_r$ are $\bar{\nu}$-independent polynomials; moreover, by Lemma 4.3.1 there is a morphism between the span of the set $\{1\} \cup \left\{(X - x_k)^{-l}\right\}_{k,l \in \bar{\nu}_{j,k}}$ and $\mathbb{F}_{\leq \nu_j^*}[X]$. Thus, we conclude that $m_{j^*}(X) \neq 0$.

Since $c_i^*(x^*, \mathsf{coms}) = [y^*]_1$ by the check introduced in $\mathbf{H}_5$, and $c_i^*(x^*, \mathsf{coms}) = \hat{c}_i(x^*, \mathsf{coms})$ by definition, we can reduce to Aff-MDH as follows. The reduction generates the SRS and simulates using the trapdoor s, while the commitments coms are received by the Aff-MDH challenger[11]. The reduction outputs $((\mu_j)_j, \hat{y})$, where the coefficients $\mu_j \leftarrow m_j(x^*)$ and $\hat{y} = y^* - m_0(x^*)$.

Given that the *Hash check* is satisfied, $((\mathsf{b}_{i,r})_r; (A_{i,r})_r; h) \rightarrow_{\mathsf{RO}} a$, and $h(a) = x^*$, which implies that c_i^* is a function of the coefficients $c_{i,r,j}, o_{i,r,j,k,l}$ and the polynomials $A_{i,r}$ that are fixed before a (and hence x^*) is computed. By Schwartz-Zippel, we derive that the coefficient $\mu_{j^*} = m_{j^*}(x^*)$ is null only with negligible probability $\mathsf{deg}(h)(\bar{\nu} + \max_r \mathsf{deg}(A_{i,r}))/q$. □

Finally, we notice that in $\mathbf{H}_6$, $\mathcal{E}$ successfully extracts all the witness polynomials f_i, for $i \in \mathcal{I}$. Then we conclude that $\epsilon_6 = 0$. □

4.5.2 Simulation Extractability of the Linearization Trick

In this section we formalize the linearization trick for KZG commitments [GWC19, OL] as a CP-SNARK for the relation $\mathcal{R}_{\mathsf{lin}}$ that upon instance:

$$\mathbb{x} \coloneqq ((\mathsf{c}_j)_{j \in [m]}, (\mathsf{b}_i)_{i \in [n]}, (G_i)_{i \in [n]}, x, y),$$

whose witness $\mathbb{w} = (C_j)_{j \in [m]}, (B_i)_{i \in [n]}$ are polynomials committed in the instance, and that outputs 1 if and only if

$$\sum_{i=1}^{n} A_i(x) B_i(x) = y,$$

[11] In particular, this means that the commitments are sampled from $\mathcal{D}$, which is identically distributed to $\tilde{\mathcal{D}}$, as argued in $\mathbf{H}_2$.

and where $A_i(X) \coloneqq G_i((C_j(X))_j, X)$ with $G_i \in \mathbb{F}[X_1, \ldots, X_m, X]$.

We call the polynomials C_j (resp. commitments c_j) the *core polynomials* (resp. commitments); moreover, we call the polynomials A_i and B_i (resp. the commitments b_i) the *left* and *right polynomials* (resp. commitments).

We define $\mathsf{CP}_{\mathsf{lin}}$ that uses $\mathsf{CP}_{\mathtt{m\text{-}evl}}$ as inner scheme:

$\mathsf{Prove}_{\mathsf{lin}}(\mathsf{ek}, \mathbb{x}, \mathbb{w})$**:** compute $\pi_{\mathtt{m\text{-}evl}} \leftarrow \mathsf{Prove}_{\mathtt{m\text{-}evl}}(\mathbb{x}_{\mathtt{m\text{-}evl}}, ((C_j)_j, R))$, where $R(X) \coloneqq \sum_i A_i(x)B_i(X)$, $\mathsf{r} \coloneqq \sum_i A_i(x)\mathsf{b}_i$, and $\mathbb{x}_{\mathtt{m\text{-}evl}} \coloneqq (x, (\mathsf{c}_j, C_j(x))_j, (\mathsf{r}, y))$. Output $\pi \coloneqq (\pi_{\mathtt{m\text{-}evl}}, (C_j(x))_j)$

$\mathsf{Verify}_{\mathsf{lin}}(\mathsf{vk}, \mathbb{x}, \pi)$**:** parse π as $(\pi_{\mathtt{m\text{-}evl}}, (y_j)_j)$, compute r as $\sum_i G_i((y_j)_j, x)\mathsf{b}_i$. Output $\mathsf{Verify}_{\mathtt{m\text{-}evl}}(\mathsf{vk}, \mathbb{x}_{\mathtt{m\text{-}evl}}, \pi_{\mathtt{m\text{-}evl}})$, where $\mathbb{x}_{\mathtt{m\text{-}evl}} \coloneqq (x, ((\mathsf{c}_j, y_j)_j, (\mathsf{r}, y)))$

This scheme is not zero-knowledge as the proofs leak some information on the witness, that are the values $y_j = C_j(x)$. Formally, it achieves L_{lin}-leaky zero-knowledge where $L_{\mathsf{lin}}(\mathbb{x}, \mathbb{w}) \coloneqq (\mathbb{w}.C_j(\mathbb{x}.x))_j$. We define the simulator $\mathcal{S} \coloneqq (\mathcal{S}_0, \mathcal{S}_1)$, where $\mathcal{S}_0$ outputs the trapdoor information s together with the srs, and $\mathcal{S}_1$ simulates proofs for $\mathbb{x} \coloneqq ((\mathsf{c}_j)_{j\in[m]}, (\mathsf{b}_i)_{i\in[n]}, (G_i)_{i\in[n]}, x, y)$ and leakage $(y_j)_{j\in[m]}$ computing $\pi_{\mathtt{m\text{-}evl}} \coloneqq (s - x)^{-1}(\sum_j \rho^{i-1}(\mathsf{c}_j - [y_j]_1) + \rho^m(\mathsf{r} - y))$, where $\rho \coloneqq \mathsf{RO}(\mathsf{batch}\|x, (\mathsf{c}_j, C_j(x))_j, (\mathsf{r}, y))$, $\mathsf{r} \coloneqq \sum_i A_i(x)\mathsf{b}_i$ and outputs the proof $\pi \coloneqq (\pi_{\mathtt{m\text{-}evl}}, (y_j)_j)$.

The extraction policy. Let $\Phi_{\mathsf{lin}}^{\mathcal{J},\nu}$ be the policy parametrized by $\nu \in \mathbb{N}$ and $\mathcal{J} \subseteq [n]$, for $n \in \mathbb{N}$, described below:

Hash Check (for the linearization trick): parse the forged instance $\mathbb{x}^* \coloneqq ((\mathsf{c}_j^*)_j, (\mathsf{b}_i^*)_i, (G_i^*)_i, x^*, y^*)$, return 1 if and only if there exists a polynomial h such that:

- $((\mathsf{c}_j^*)_j, (\mathsf{b}_i^*)_i; (G_i^*)_i, h) \to_{\mathsf{RO}} a$ and $h(a) = x^*$;
- $\forall j : \nu > \sum_k \max_{\pi\in\mathsf{proofs}} \nu_\pi(j, k)$ where proofs is the list of simulated proofs.

Partial-Extraction Check: parse $\mathsf{aux}_{\mathcal{E}}$, find polynomials $(B_i^*)_{i\in\mathcal{J}}$ and return 1 if and only if b_i^* commits to B_i^*, $\forall i \in \mathcal{J}$.

Definition 4.5.8. *Let $\mathbf{\Phi}_{\mathsf{lin}}^{\mathcal{J},\nu}$ be the set of policies $\Phi_{\mathcal{D}} = (\Phi_0^{\mathcal{D}}, \Phi_{\mathsf{lin}}^{\mathcal{J},\nu})$ for a distribution $\mathcal{D}$ where:*

- *$\Phi_0^{\mathcal{D}}$ on input group parameters $\mathsf{pp}_{\mathbb{G}}$ outputs $\mathsf{pp}_\Phi \coloneqq \mathsf{coms}$, where coms is a vector of commitments sampled from $\mathcal{D}$.*
- *$\mathcal{D}$ is witness sampleable and the $\mathcal{D}$-Aff-MDH assumption holds.*

The *Partial-Extraction Check* allows to define the concept of *partial extractability* (see [BCF+21, CFH+22]) within the framework of Φ-simulation extractability. The definition of partial extractability allows the adversary to provide to the extractability experiment one part of the witness, while the extractor must find the remaining part. Looking ahead, this check allows defining more flexible notions of extractability, for example, PLONK's verifier needs to check two linearization trick instances on a non-disjunct set of polynomials, thus we can partition the polynomials to extract between the two instances and, in doing so, we can loosen the independence requirements from the two instances. We give more details in Section 4.6.4.

To formalize the extractability of the linearization trick we crucially rely on the framework of $\mathcal{F}$-extractability. In particular, we consider the function $\mathcal{F}_{\mathcal{J},\nu}(\mathbb{w})$, for parameters $\mathcal{J} \subseteq [n]$ and $\nu \in \mathbb{N}$, that parses $\mathbb{w}$ as $(C_j)_j, (B_i)_i$, computes for all i the polynomial $A_i(X) \coloneqq G_i((C_j(X))_j, X)$, and outputs $\mathbb{w}$ if $(A_i)_{i\notin\mathcal{J}}$ are ν-independent, otherwise outputs only $(C_j^*)_j$.

The $\mathcal{F}_{\mathcal{J},\nu}$-extractability and the *Hash Check* go hand in hand, the former specifies the condition under which extraction of the right polynomials can happen while the latter sets the rules, for the adversary, so that such condition holds.

Theorem 4.5.2. *For any $n, \nu \in \mathbb{N}, \mathcal{J} \subseteq [n]$, $\mathsf{CP}_{\mathsf{lin}}$ is $(\mathbf{\Phi}_{\mathsf{lin}}^{\mathcal{J},\nu}, \mathcal{F}_{\mathcal{J},\nu})$-simulation-extractable in the AGM under the OMSDH assumption.*

Proof Intuition. Thanks to the heavy lifting of Theorem 4.5.1 the proof of Theorem 4.5.2 is not much different than a proof of (standard) extractability in the AGMOS [LPS23] would be. In fact, the proof can be summarized as two direct reductions to the SE of $\mathsf{CP}_{\mathtt{m\text{-}evl}}$. In the first reduction, which is almost straight-forward, we show how to extract the core polynomials. On the other hand, the second reduction needs a careful analysis as, in fact, the extractor of $\mathsf{CP}_{\mathtt{m\text{-}evl}}$ extracts $R(X) = \sum A_i(x^*)B_i(X)$ while we need to show how to extract the polynomials $(B_i(X))_i$. For simplicity, assume that the adversary obtains an obliviously sampled element c, thus we can write $\mathsf{b}_i = [B_i(s)]_1 + \bar{B}_i(s) \cdot \mathsf{c}$. We need to show that $\bar{B}_i \equiv 0$, and we can assume, thanks to the SE of $\mathsf{CP}_{\mathtt{m\text{-}evl}}$, that $\sum A_i(x^*)\bar{B}_i(X) \equiv 0$. In proving knowledge extractability, we can just rely on the linear independence of the polynomials A_i and the Schwartz-Zippel lemma, for simulation extractability we additionally use the ν-independence and the second item of the Hash Check property.

Removing the Hash Check. $\Phi_{\mathsf{lin}}^{\mathcal{J},\nu}$ is a sufficient ingredient of our compiler to prove the simulation extractability of zkSNARKs such as PLONK or Marlin: as we explain in Section 4.6, in these two protocols the verifier checks that the polynomials sent by the prover satisfy some predicate on some *random* points, which allows us to reuse the proofs of an adversary to the SNARKs in the reduction to the simulation extractability of $\mathsf{CP}_{\mathsf{lin}}$ since they match the *Hash check*. Looking ahead, we call these checks *focal* as they play an important role in our compilation strategy.

However, there may be other protocols that involve also checking equations over some fixed, or not sufficiently random, points. In this paragraph, we show that we can prove $\mathsf{CP}_{\mathsf{lin}}$ simulation extractable even when the Hash Check is not satisfied, as long as the adversary is able to produce a proof algebraic inconsistent w.r.t. view. This allows us to enlarge the class of protocols that our compiler to zkSNARKs captures. A similar result was proved for the scheme $\mathsf{CP}_{\mathtt{m\text{-}evl}}$ in [FFK+23]. Let $\Phi_{\mathsf{lin}+}$ be the policy that performs the following check:

- **Algebraic Check:** let $\mathbb{x} := ((\mathsf{c}_j)_j, (\mathsf{b}_i)_i, (G_i)_i, x, y)$ and $\pi := (\hat{\pi}, (y_j)_j)$, returns 1 if and only if there exists a tuple $(\mathbb{x}' = ((\mathsf{c}_j)_j, (\mathsf{b}_i)_i, (G_i)_i, x, y'), \pi' = \hat{\pi}', (y'_j)_j)$ in $\mathcal{Q}_{\mathsf{sim}}$ such that

$$y' \neq y \vee \exists j : y_j \neq y'_j \tag{4.4}$$

Theorem 4.5.3. *$\mathsf{CP}_{\mathsf{lin}}$ is $(\Phi_{\mathsf{lin}+}, id)$-simulation-extractable in the AGM under the OMSDH assumption.*

Proof. We show a reduction $\mathcal{B}$ to the $(\mathbf{\Phi}_{\mathsf{lin}}^{\emptyset,0}, id)$-simulation extractability of $\mathsf{CP}_{\mathsf{lin}}$. For the simulation, the reduction simply proxies all the queries back and forth between the adversary and the challenger. At forgery time, $\mathcal{B}$ finds the commitments $(\mathsf{c}_j)_{j\in[m]}$, $(\mathsf{b}_i)_{i\in[n]}$ in the forgery instance $\mathbb{x}$, and derives the linearization commitment r as the verifier would do when verifying the forgery proof π, namely using G_i, x and the field elements y_j contained in it and the commitments b_i. Also, let ρ be the batch coefficient used by the verifier in this step, and let y'_j

be the field elements contained in the simulated proof π' that is not algebraic consistent with π: if the adversary complies with the extraction policy, this proof exists; this means that, for all j, $\mathcal{B}$ can find in π' the values y'_j such that the pairs (c_j, y'_j) are algebraic consistent with the current view view; $\mathcal{B}$ can also find y'_r such that (r, y'_r) is algebraic consistent with view. Then $\mathcal{B}$ queries $\mathcal{S}_1$ to get (single-eval) proofs $\bar{\pi}_j$ and $\bar{\pi}_r$ associated with the pairs defined above, and defines

$$\bar{\pi} \coloneqq \sum_j \rho^{j-1}\bar{\pi}_j + \rho^m \bar{\pi}_r$$

If $\bar{\pi} = \pi$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ submits the instance forgery $\mathbb{x}^* \coloneqq ((\mathsf{c}^*, [0]_1), x^*, 0)$ and the proof $\pi^* \coloneqq (\hat{\pi}^*, y^*)$ where:

$$\begin{aligned}
&\mathsf{c}^* \leftarrow (\bar{\pi} - \pi)\\
&\hat{\pi}^* \leftarrow (\bar{\pi} - \pi)(x - x^*)^{-1}\\
&y^* \leftarrow (\sum_j \rho^{j-1}(y'_j - y_j) + \rho^m(y'_r - y))(x - x^*)^{-1}
\end{aligned}$$

the forgery point $x^* \leftarrow \mathsf{RO}(s)$, and s is a string never queried to the RO and containing c^* as substring that yields $\mathsf{c}^* \rightarrow_{\mathsf{RO}} x^*$.

First, we show that the probability that $\mathcal{B}$ aborts when the adversary complies with the policy and submits a successful forgery is only negligible. Let $\pi_j \coloneqq (\mathsf{c}_j - y_j)(s - x^*)^{-1}$, and let $\pi_r \coloneqq (\mathsf{r} - y)(s - x^*)^{-1}$. We have that the proof $\pi = \sum_j \rho^{j-1}\pi_j + \rho^m \pi_r$. Also, by definition of Eq. (4.4), and because the KZG (single-eval) proofs $\bar{\pi}_j$ and $\bar{\pi}_r$ are unique, either $\pi_r \neq \bar{\pi}_r$ or there exist j such that $\bar{\pi}_j \neq \pi_j$. Since the coefficient ρ is chosen uniformly at random after the proofs of the adversary are fixed, and is also independent of the proofs $\bar{\pi}_j, \bar{\pi}_r$ of the reduction, we conclude by Schwartz-Zippel that $\pi = \bar{\pi}$ with probability at most equal to $\frac{m}{q}$.

Second, we observe that $\mathcal{B}$ complies with the policy in the extractability experiment: the Hash Check is satisfied because we define the point x^* as the output of a RO query including the commitment c^*, and moreover the Partial-Extraction Check is trivially satisfied because the reduction does not have to add any polynomial in $\mathsf{aux}_{\mathcal{E}}$.

Finally, the forgery $(\mathbb{x}^*, \pi^*)$ satisfies the verification equation. In the verification procedure, the linearization commitment is set to be $y^* \cdot [0]_1 = [0]_1$. Because of the homomorphic properties of KZG, then, the verifier has only to check that $\hat{\pi}^*$ is a valid proof for the (single-eval) statement (c^*, x^*, y^*), whose correctness follows from Eq. (4.1). □

Proof. We define our extractor $\mathcal{E}_{\mathcal{J},\nu}$ to be the extractor that, by looking at the algebraic representations, returns the polynomials $C_j(X) \coloneqq f_{\mathsf{c}_j}(X)$ for all $j \in [m]$, computes the polynomials $A_i(X)$, and if $\{A_i\}_{i \notin \mathcal{J}}$ are ν-independent, it additionally returns $B_i(X) \coloneqq f_{\mathsf{b}_i}(X)$ for all $i \in [n]$.

We let $\mathbf{H}_0$ be the $\mathbf{Exp}^{\Phi\text{-}se}_{[}\mathbf{\Phi}^{\mathcal{J},\nu}_{\mathsf{lin}}]\mathcal{A}, \mathcal{S}, \mathcal{E}$ experiment, and we denote by $\epsilon_i \coloneqq \Pr[\mathbf{H}_i = 1]$.

Hybrid $\mathbf{H}_1$. This hybrid is the same as $\mathbf{H}_0$, and it returns 0 if there exists $j \in [m]$ such that $\mathsf{c}_j \neq [C_j(s)]_1$ or $C_j(x^*) \neq y^*_j$.

Lemma 4.5.4. $\epsilon_1 \leq \epsilon_0 + \epsilon_{\texttt{m-evl}}$

Proof. Recall that $\sigma_{\mathcal{I}}$ is the $\mathcal{I}$-projection function. We reduce to the $(\mathbf{\Phi}^{\texttt{adpt}}_{\texttt{m-evl}}, \sigma_{[m]})$-simulation extractability of $\mathsf{CP}_{\texttt{m-evl}}$. We recall that the extractor of the experiment does only guarantee (if the policy is satisfied) to extract the first m polynomials.

The reduction $\mathcal{B}$ takes as input the SRS and the commitments coms from the challenger and forwards them to $\mathcal{A}$. It trivially answers the queries of $\mathcal{A}$:

- **RO-query:** Proxy the query to $\mathcal{S}_2$.
- **SIM-query:** On input an instance $\mathbb{x}_{\mathsf{lin}}$, and leakage $(y_j)_{j\in[m]}$ define the multi-eval instance $\mathbb{x}_{\mathtt{m\text{-}evl}}$ as the honest prover would do, and query $\mathcal{S}_1$ on it.

Upon forgery $(\mathbb{x}^*, \pi^*)$ from $\mathcal{A}$, where $\mathbb{x}^* = (x^*, (\mathsf{c}_j)_j, (\mathsf{b}_i)_i, y^*)$, and $\pi^* = (\pi^*_{\mathtt{m\text{-}evl}}, (y^*_j)_j)$, $\mathcal{B}$ returns a multi-eval forgery $(\mathbb{x}^*_{\mathtt{m\text{-}evl}}, \pi^*_{\mathtt{m\text{-}evl}})$ where the statement is $\mathbb{x}^*_{\mathtt{m\text{-}evl}} = (x^*, ((\mathsf{c}_j)_j, \mathsf{r}), ((y^*_j)_j, y^*))$, with $\mathsf{r} = \sum_i G_i((y_j)_j, x)\mathsf{b}_i$, and $\pi^*_{\mathtt{m\text{-}evl}}$ is the proof in π^*.

Notice that this forgery passes the *Hash check* predicate for $\mathsf{CP}_{\mathtt{m\text{-}evl}}$ when the forgery of $\mathcal{A}$ passes the *Hash check* for $\mathsf{CP}_{\mathsf{lin}}$: in particular, for some polynomial h we have that, for all $j \in [m]$, $(\mathsf{c}_j; h) \to_{\mathsf{RO}} a$, and $h(a) = x^*$. We have that the (canonical) extractor for $\mathsf{CP}_{\mathtt{m\text{-}evl}}$ would successfully extract, unless with probability $\epsilon_{\mathtt{m\text{-}evl}}$, all the witness polynomials $C_j(X)$ associated with the commitments c_j and such that $C_j(x^*) = y^*_j$. □

Hybrid $\mathbf{H}_2$. This hybrid is the same as $\mathbf{H}_1$, except it returns 0 if $\{A_i\}_{i\notin\mathcal{J}}$ are ν-independent polynomials and:

- $\sum_{i\in[n]} A_i(x^*)B_i(x^*) \neq y^*$
- or there exists $i \in [n]$ such that $\mathsf{b}_i \neq [B_i(s)]_1$

Lemma 4.5.5. $\epsilon_2 \leq \epsilon_1 + \epsilon_{\mathsf{Aff\text{-}MDH}} + \epsilon_{\mathtt{m\text{-}evl}} + \frac{\mathtt{deg}(h)(\max_i \mathtt{deg}(A_i)+\bar{\nu})}{q}$

Proof. Notice that for all $i \in \mathcal{J}$, $\mathsf{b}_i = [B_i(s)]_1$, where the polynomials $(B_i)_{i\in\mathcal{J}}$ are output by the adversary itself, by definition of the *Partial-Extraction Check* in the extraction policy. If the distinguishing event occurs, namely $\{A_i\}_{i\notin\mathcal{J}}$ are ν-independent and $\sum_{i\in[n]} A_i(x^*)B_i(x^*) \neq y^*$ or $\mathsf{b}_i \neq [B_i(s)]_1$ for $i \notin \mathcal{J}$, we can reduce to the simulation extractability of $\mathsf{CP}_{\mathtt{m\text{-}evl}}$ as follows.

The reduction $\mathcal{B}$ simulates the experiment for $\mathcal{A}$ as in the reduction described in Lemma 4.5.4. Then, upon forgery $(\mathbb{x}^*, \pi^*)$ from $\mathcal{A}$, where $\mathbb{x}^* = ((\mathsf{c}_j)_j, (\mathsf{b}_i)_i, (G^*_i)_i, x^*, y^*)$, and $\pi^* = (\pi^*_{\mathtt{m\text{-}evl}}, (y^*_j)_j)$, $\mathcal{B}$ computes the values:

$$q_j \coloneqq \left[(C_j(s) - C_j(x^*))(s - x^*)^{-1}\right]_1, \forall j \in [m]$$
$$q_{m+1} \coloneqq \sum_{i\in\mathcal{J}} \left[(A_i(s)B_i(s) - A_i(x^*)B_i(x^*))(s - x^*)^{-1}\right]_1.$$

The adversary $\mathcal{B}$ sets as forgery the statement $\mathbb{x}'_{\mathtt{m\text{-}evl}} \coloneqq (x^*, \mathsf{r}', y')$ and proof $\pi'_{\mathtt{m\text{-}evl}}$, where:

$$\mathsf{r}' \leftarrow \sum_{i\in[n]} A_i(x^*)\mathsf{b}_i - \sum_{i\in J} [A_i(s)B_i(s)]_1$$
$$y' \leftarrow y^* - \sum_{i\in\mathcal{J}} A_i(x^*)B_i(x^*)$$
$$\pi'_{\mathtt{m\text{-}evl}} \leftarrow \rho^{-m}(\pi^*_{\mathtt{m\text{-}evl}} - \sum_{j\in[m+1]} \rho^{j-1} q_j)$$

and $\rho \leftarrow \mathsf{RO}(\mathsf{batch}\|\mathbb{x}^*_{\mathtt{m\text{-}evl}})$. Notice that the above forgery is for a multi-eval of size 1, namely it is a single-eval forgery, and thus the batch coefficient $\rho' \leftarrow \mathsf{RO}(\mathsf{batch}\|\mathbb{x}'_{\mathtt{m\text{-}evl}})$ is actually never used by the verifier to check the proof: this is why it passes the verification equation when the forgery of $\mathcal{A}$ satisfies the verification equation.

Differently from the reduction in Lemma 4.5.4, the extractor of $\mathsf{CP}_{\mathtt{m\text{-}evl}}$ can extract only one witness, i.e., the polynomial committed in r'.

If the *Hash check* for $\mathsf{CP}_{\mathsf{lin}}$ is satisfied, so does the *Hash check* for $\mathsf{CP}_{\mathtt{m\text{-}evl}}$: in particular, by definition of the distinguishing event, the polynomials $(A_i(X))_{i\notin\mathcal{J}}$ are ν-independent. We have that, unless with probability $\epsilon_{\mathtt{m\text{-}evl}}$, the canonical extractor of $\mathsf{CP}_{\mathtt{m\text{-}evl}}$ would extract from $\mathsf{r}' \coloneqq \sum_{i\notin\mathcal{J}} A_i(x^*)\mathsf{b}_i$ the polynomial

$$R'(X) \coloneqq \sum_{i\notin\mathcal{J}} A_i(x^*)B_i(X)$$

such that $R'(x^*) = y'$.

Similarly to the proof of Theorem 4.5.1 (just before Hybrid $\mathbf{H}_5$), for all $i \notin \mathcal{J}$ we can associate the commitment b_i with a polynomial equal to $B_i(X)+\tilde{B}_{i,0}(X)+\sum_j Y_j\tilde{B}_{i,j}(X)$, where $\tilde{B}_{i,0}(X)$ depends only on the simulated proofs, while $\tilde{B}_{i,j}(X)$ depends on the simulated proofs and simulated commitment c_j, and we can associate the commitment r' with a polynomial $R'(X,\vec{Y})$ such that $\mathsf{r}' = R'(s,\mathsf{coms})$ and $R'(X,\vec{Y})$ is equal to $M_0(X) + \sum_j M_j(X)Y_j$, where:

$$\begin{aligned} M_0(X) &= \sum_{i\notin\mathcal{J}} A_i(x^*)B_i(X) + \sum_{i\notin\mathcal{J}} \tilde{B}_{i,0}(X), \\ M_j(X) &= \sum_{i\notin\mathcal{J}} A_i(x^*)\tilde{B}_{i,j}(X), \quad \forall j > 0, \end{aligned}$$

and it holds that:

1. $\tilde{B}_{i,0}(X) \equiv 0$ if for all $j > 0 : \tilde{B}_{i,j} \equiv 0$.
2. For all i, j, $\tilde{B}_{i,j}$ is an element of a space isomorphic to $\mathbb{F}_{\leq\nu}[X]$.
3. $R'(x^*, \mathsf{coms}) = [y']_1$.

To see Item 1, notice that a simulated proof for (c, x, y) is equal to $\mathsf{c}/(s-x) + [-y/(s-x)]_1$. The rational function $\tilde{B}_{i,0}(X)$ accounts the second addends $-y/(X-x)$ from all the simulated proofs while the $\tilde{B}_{i,j}$ for $j > 0$ take care of the remaining addends. If the latter rational functions are 0 then it means that the adversary did not query the simulation oracle and therefore also the $\tilde{B}_{i,0}$ are 0 polynomials.

When the *Hash Check* holds the polynomials $\tilde{B}_{i,j}$ are fixed before x^* is computed by the RO, similarly to the proof of Theorem 4.5.1, these polynomials are fixed by the algebraic representation of the commitment r. If $\mathsf{r} \to_{\mathsf{RO}} x^*$, then the claim holds.

First, we notice that if for all $i \notin \mathcal{J}$ and for all j, the polynomial $\tilde{B}_{i,j} \equiv 0$, then $R'(X,\vec{Y}) = R'(X)$. We derive that

$$\sum_{i\in[n]} A_i(x^*)B_i(x^*) = \sum_{i\in\mathcal{J}} A_i(x^*)B_i(x^*) + R'(x^*) = y^*$$

We now bound the probability that there exist indexes i, j such that $\tilde{B}_{i,j} \not\equiv 0$. Assume, to reach a contradiction, that there exist i^*, j^* such that $\tilde{B}_{i^*,j^*} \not\equiv 0$. First, we notice that, by Item 1, we can assume $j^* > 0$. Second, we notice that $M_{j^*} \not\equiv 0$ because $\{A_i\}_{i\notin\mathcal{J}}$ are ν-independent by definition, and, by Item 2, for all i, j $\tilde{B}_{i,j}$ is an element of a space isomorphic to $\mathbb{F}_{\leq\nu}[X]$. More in detail, let $\hat{M}_{j^*}(X) = \sum_i A_i(X)B_i(X)$, because of the ν-independence and

the bound on the degree of the $\tilde{B}_{i,j}$ we have that $\hat{M}_{j^*}(X) \not\equiv 0$. Now assume $M_{j^*}(X) \equiv 0$, then $M_{j^*}(x^*) = 0$ and thus $\hat{M}_{j^*}(x^*) = 0$. However, notice that $\hat{M}_{j^*}$ is defined by the forged instance and therefore before x^* is sampled, which means that applying the Schwartz-Zippel lemma $M_{j^*}(X) \equiv 0$ only with probability:

$$\mathsf{deg}(M_{j^*} \circ h)/q = \mathsf{deg}(h)(\max_i \mathsf{deg}(A_i) + \bar{\nu})/q.$$

When $M_{j^*} \not\equiv 0$, we can reduce to Aff-MDH. The reduction generates the SRS and simulates using the trapdoor s, while the commitments coms are received by the Aff-MDH challenger. The reduction to Aff-MDH outputs $((\mu_j)_j, \hat{y})$, where the coefficients $\mu_j \leftarrow m_j(x^*)$ and $\hat{y} = y' - m_0(x^*)$. Item 3 implies the correctness of the forgery of the reduction to Aff-MDH.

Putting together, the distinguishing event implies either a forgery for $\mathsf{CP}_{\mathtt{m\text{-}evl}}$ or a forgery for the Aff-MDH assumptions, therefore the statement of the lemma follows. □

Finally, we notice that $\epsilon_2 = 0$ because the extractor returns the witness polynomials C_j, for all $j \in [m]$, by the change introduced in $\mathbf{H}_1$. Also, if $\{A_i\}_{i \notin \mathcal{J}}$ are ν-independent, then it also extracts valid witnesses B_i, for all $i \in [n]$, because of the change introduced in $\mathbf{H}_2$. □

4.6 Generalizing Polynomial Interactive Oracle Proofs

We generalize PIOPs by allowing the verifier's queries to be (arbitrary) predicates over the prover's oracles. To this end, we use the formalism of oracle relations introduced in [CBBZ23]. Roughly speaking, an oracle relation could be seen as the *oracle-world* counterpart of commit-and-prove relation. In particular, as we use them in the next definition, oracle relations are a useful abstraction which allows defining predicates over the oracles sent by the prover in the execution of a PIOP.

Definition 4.6.1 (Oracle Relations, [CBBZ23])**.** *An oracle (indexed) relation $\mathcal{R}$ is an (indexed) relation when the instances $\mathbb{x}$ of $\mathcal{R}$ contain pointers to oracle polynomials over some field $\mathbb{F}$. The actual polynomials corresponding to the oracles are contained in the witness. We denote the pointer to the oracle polynomial f by $[\![f]\!]$, let $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$ we denote with $\mathsf{oracles}(\mathbb{x}) = \{[\![f_1]\!], [\![f_2]\!], \ldots, [\![f_k]\!]\}$ for some k the pointers to the polynomial oracles in $\mathbb{x}$ and $\mathbb{w} = (f_1, f_2, \ldots, f_k)$.*

Definition 4.6.2 ((Holographic) $\hat{\mathcal{R}}$-PIOP)**.** *Let $\mathcal{F}$ be a family of finite fields, let $\mathcal{R}$ be an oracle indexed relation and $\hat{\mathcal{R}}$ be an oracle relation. A (public-coin non-adaptive) Holographic $\hat{\mathcal{R}}$-PIOP over $\mathcal{F}$ for $\mathcal{R}$ is a tuple $\mathsf{PIOP} := (r, n, m, D, \mathsf{I}, \mathsf{P}, \mathsf{V})$ where $r, n, m, D \colon \{0,1\}^* \to \mathbb{N}$ are polynomial-time computable functions, and $\mathsf{I}, \mathsf{P}, \mathsf{V}$ are three algorithms for the indexer, prover and verifier respectively, that work as follows.*

Offline phase: *The indexer $\mathsf{I}(\mathbb{F}, \mathbb{i})$ is executed on input a field $\mathbb{F} \in \mathcal{F}$ and a relation description $\mathbb{i}$, and it returns $n(0)$ polynomials $\{p_{0,j}\}_{j \in [n(0)]}$ encoding the relation $\mathbb{i}$.*

Online phase: *The prover $\mathsf{P}(\mathbb{F}, \mathbb{i}, \mathbb{x}, \mathbb{w})$ and the verifier $\mathsf{V}^{\mathsf{I}(\mathbb{F},\mathbb{i})}(\mathbb{F}, \mathbb{x})$ are executed for $r(|\mathbb{i}|)$ rounds; the prover has a tuple $(\mathbb{F}, \mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$ and the verifier has an instance $\mathbb{x}$ and oracle access to the polynomials encoding $\mathbb{i}$.*

In the i-th round, P sends $m(i)$ messages $\{\pi_{i,j} \in \mathbb{F}\}_{j \in [m(i)]}$, and $n(i)$ oracle polynomials $\{[\![p_{i,j}]\!] : p_{i,j} \in \mathbb{F}[X]\}_{j \in [n(i)]}$ of degree at most $D := D(|\mathbb{i}|)$, while V replies (except for the last round) with a uniformly random message $\rho_i \in \mathbb{F}$.

Decision phase: *Let* $r := r(|\mathbb{i}|)$, $n := \sum_{k=0}^{r} n(k)$, $m := \sum_{k=1}^{r} m(k)$. *After the* r*-th round, let the verifier* $\mathcal{V}^{\mathsf{I}(\mathbb{F},\mathbb{i})}(\mathbb{F}, \mathbb{x}, \vec{\pi}, \vec{\rho})$*, on input the description of the field* $\mathbb{F}$*, the verifier messages* $\vec{\rho} := (\rho_1, \ldots, \rho_{r-1})$*, the messages of the prover* $\vec{\pi} := (\pi_1, \ldots, \pi_m)$ *outputs the instance* $\hat{\mathbb{x}}$ *with* $\mathsf{oracles}(\hat{\mathbb{x}}) \subseteq \{[\![p_1]\!], \ldots, [\![p_n]\!]\}$*. The verifier accepts if* $\hat{\mathbb{x}} \in \mathcal{L}_{\hat{\mathcal{R}}}$.

We simply say that PIOP is an r-rounds PIOP if the number of rounds is constant and independent of the size of the index. We give some additional notation. In the following, we will use two different ways to index (the pointers to) the oracle polynomials in a PIOP protocol's execution. We will refer to the oracle polynomials sent by the prover and by the indexer either as $[\![p_{i,j}]\!]$, with double indexes, or as the $[\![p_k]\!]$, with a single index, where $k = \sum_{i'=1,\ldots,i-1} n(i') + j$. We define the oracle index $\mathsf{index}([\![f]\!])$ as the index in the transcript associated with (the pointer to) the polynomial oracle $[\![f]\!]$. Similarly to the set $\mathsf{oracles}(\hat{\mathbb{x}})$, we define the set $\mathsf{indexes}(\hat{\mathbb{x}}) := \{\mathsf{index}([\![f]\!]) : [\![f]\!] \in \mathsf{oracles}(\hat{\mathbb{x}})\}$ the indexes in the transcript associated with (the pointers to) the polynomial oracles involved in $\hat{\mathbb{x}}$.

Simulation-friendly polynomial oracles. Hereafter, we introduce the notion of simulation-friendly polynomial oracles to abstract how our compiler generates instance-independent commitments for the oracles sent during a PIOP protocol's execution.

Definition 4.6.3 (PIOP with simulation-friendly polynomial oracles)**.** *A PIOP* PIOP *has simulation-friendly polynomial oracles if for every* $\mathbb{F}$ *and* $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$ *the distribution* $(\mathsf{Com}(\mathsf{ck}, p_i))_i$ *is computationally indistinguishable from the uniform distribution over* $\mathcal{C}^n$ *where* $\mathcal{C}$ *is the commitment space and where* $(p_i)_i$ *are the oracles sent by the prover* $\mathsf{P}(\mathbb{F}, \mathbb{i}, \mathbb{x}, \mathbb{w})$ *in the interaction with* $\mathcal{V}(\mathbb{F}, \mathbb{x})$.

If the commitment scheme is hiding then this property is trivially true. For the case of non-hiding commitments, one may rely on Decisional Uber Assumption [Boy08] that reduces to discrete log for algebraic adversaries [RS20].

Verifier Checks. It is often the case that the relation $\hat{\mathcal{R}}$, for an $\hat{\mathcal{R}}$-PIOP, is the logical conjunction of a (sub)relation. In this case, we consider $\hat{\mathbb{x}} := (\hat{\mathbb{x}}_k)_k$ and the verifier returns 1 when all the checks $\hat{\mathbb{x}}_i$ are satisfied. When looking at concrete examples of PIOPs, in the rest of this section, we will assume that this natural approach is used by the verifier: for sake of simplicity, we extend Definition 4.6.2 of an $\hat{\mathcal{R}}$-PIOP and allow the verifier to output n_e checks $(\hat{\mathbb{x}}_k)_k$, and the verifier accepts if and only if $\hat{\mathbb{x}}_k \in \mathcal{L}_{\hat{\mathcal{R}}}$ for all $k \in [n_e]$.

PIOPs with Delegation. There are cases in which the PIOP can be thought of as a two-phase protocol, sharing the same indexer I where: (i) in the first phase of the protocol, the prover P_1 takes as input the field $\mathbb{F}$, the index $\mathbb{i}$, the instance $\mathbb{x}$ and the witness $\mathbb{w}$, and interacts for a certain number of rounds with the verifier, while (ii) in the second phase, the prover P_2 that, crucially, does not take as input the witness $\mathbb{w}$, interacts with the verifier for only two rounds.[12] Since we require the output of P_2 to be uniquely determined by its input (which is also computable by an "inefficient" verifier), we call this last (witness-independent) phase a *delegation phase*.

The reason to add this new definition is to enlarge the class of PIOPs for which the technical condition in [FFK+23] (sufficient to prove Simulation Extractability of the compiled SNARK) holds.

[12] We could consider a more general setting with multiple delegation rounds; however, all the optimized constructions we are aware of only require two.

Definition 4.6.4 (Delegation Phase for a PIOP)**.** *Let* PIOP *be an* $r+1$*-rounds* $\hat{\mathcal{R}}$*-PIOP over* $\mathcal{F}$ *for* $\mathcal{R}$*. We say that* PIOP *is* $\hat{\mathcal{R}}$*-PIOP with delegation phase if we can parse* P *(resp.* $\mathcal{V}$*) as* P_1 *and* P_2 *(resp. as* $\mathcal{V}_1$ *and* $\mathcal{V}_2$*) such that there exists a verifier* $\tilde{\mathcal{V}}$ *taking as additional input the index* $\mathtt{i}$ *where (1)* $\mathsf{PIOP}_1 = (\mathsf{P}_1, \tilde{\mathcal{V}})$ *is a* $(r-1)$*-rounds* $\hat{\mathcal{R}}$*-PIOP over* $\mathcal{F}$ *for* $\mathcal{R}$ *and the queries of* $\tilde{\mathcal{V}}$ *and* $\mathcal{V}_1$ *are identical for any inputs, (2)* $\mathsf{PIOP}_2 = (\mathsf{P}_2, \mathcal{V}_2)$ *is a 2-rounds* $\hat{\mathcal{R}}$*-PIOP over* $\mathcal{F}$ *for the (P-)language of strings* $(\mathbb{F}, \mathtt{i}, (\mathtt{x}, (\vec{\pi}_j)_{j\in[r]}, (\rho_j)_{j\in[r-1]}))$ *where* $\tilde{\mathcal{V}}(\mathbb{F}, \mathtt{i}, \mathtt{x}, (\vec{\pi}_j)_{j\in[r]}, (\rho_j)_{j\in[r-1]})) = 1$ *assuming that the* $\tilde{\mathcal{V}}$*'s queries to* $\hat{\mathcal{R}}$ *are answered positively.*

Uniqueness of delegation phase. *Moreover, we have that for all* $\mathbb{F}, \mathtt{i}, \mathtt{x}, (\vec{\pi}_j)_{j\in[r]}, (\rho_j)_{j\in[r-1]}$ *the probability, taken over the* $\mathcal{V}_2$*'s message* $\rho_r \leftarrow\!\!\$\ \mathbb{F}$*, that* $\mathcal{V}_2$ *on input* $(\mathbb{F}, \mathtt{x}, (\vec{\pi}_j)_{j\in[r]}, (\rho_j)_{j\in[r-1]})$ *accepts on two transcripts, that are different in the first tuple of messages and polynomials, is negligible in* $\log|\mathbb{F}|$*.*

In the following, we simply refer to an *r-rounds* $\hat{\mathcal{R}}$*-PIOP with a Delegation Phase*, denoting it as $\mathsf{PIOP}_1 \| \mathsf{PIOP}_2$, as the $(r+1)$-rounds $\hat{\mathcal{R}}$-PIOP in which the prover P first runs P_1 and interacts with the verifier $\mathcal{V}_1$ for r rounds, then runs P_2 in the last phase, while the verifier outputs the checks of $\mathcal{V}_1$ and $\mathcal{V}_2$, and accepts if and only if all the checks are satisfied.[13] Note, we say that a PIOP with delegation has simulation-friendly polynomial oracles if so does PIOP_1.

4.6.1 Polynomial $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOP

Similarly to Section 4.5.2, let $\hat{\mathcal{R}}_{\mathsf{lin}}$ be the oracle indexed relation that upon an instance

$$\hat{\mathtt{x}} := ((\llbracket c_j \rrbracket)_{j\in[m]}, (\llbracket b_i \rrbracket)_{i\in[n]}, (G_i)_{i\in[n]}, x, y),$$

outputs 1 if and only if $\sum_i a_i(x)b_i(x) = y$, where $\forall i\colon\ a_i(X) := G_i(c_1(X), \ldots, c_m(X), X)$. We refer to the polynomial oracles $(\llbracket c_j \rrbracket)_j$ as the *core polynomial oracles*, while the polynomial oracles $(\llbracket a_i \rrbracket)_i$ and $(\llbracket b_i \rrbracket)_i$ as the *left* and *right polynomial oracles* respectively. We use the shorthand $\hat{\mathtt{x}}.a_i$ to refer to the a_i defined above.

Below we formalize a class of $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOPs in which each evaluation point x chosen by the verifier for a $\hat{\mathcal{R}}_{\mathsf{lin}}$ query is a function $x = \tilde{v}(\vec{\rho})$ of its random coins, where $\tilde{v}$ is a polynomial that can be defined by the verifier depending only on the index $\mathtt{i}$ and the instance $\mathtt{x}$. The $\hat{\mathcal{R}}_{\mathsf{lin}}$ checks relying on a $\tilde{v}$ which is non-constant in the $r-1$-th random coin are called "focal", as they have a focal role to ensure extractability.

Definition 4.6.5 (Structured $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOP and focal checks)**.** *An r-rounds* $\hat{\mathcal{R}}_{\mathsf{lin}}$*-PIOP* PIOP *is* structured *if there exists a deterministic PT algorithm* $\tilde{\mathcal{V}}$ *such that for all* $\mathtt{i}$*,* $\mathtt{x}$*,* $\vec{\pi}$*,* $\vec{\rho}$*,* $k \in [n_e]$ *we have that* $\tilde{v}_k(\vec{\rho}) = \hat{\mathtt{x}}_k.x$*, where* $(\tilde{v}_k)_k \leftarrow \tilde{\mathcal{V}}(\mathbb{F}, \mathtt{i}, \mathtt{x})$ *and* $(\hat{\mathtt{x}}_k)_k \leftarrow \mathcal{V}^{\mathsf{I}(\mathbb{F},\mathtt{i})}(\mathbb{F}, \mathtt{x}, \vec{\pi}, \vec{\rho})$*.*

If $\mathsf{deg}_{r-1}(\tilde{v}_k) \geq 1$ *we say that the check* $\hat{\mathtt{x}}_k$ *is* focal*. We denote by* $\mathcal{K}_f$ *the set of all indexes* k *such that* $\hat{\mathtt{x}}_k$ *is focal.*

Finally, we introduce the notion of compilation-safeness for $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOPs. The idea of the definition below is that focal checks can be ordered in such a way that we can incrementally extract all the polynomials, starting from the trivially extractable polynomials, namely the index polynomials, and using the partial extractability property derived from Definition 4.5.8.

[13] The prover can send all the messages of the first round of PIOP_2 on the r-th round of PIOP_1, thus yielding an $r+1$ (rather than $r+2$) rounds protocol.

Definition 4.6.6 (Compiler-safe $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOP)**.** *An r-rounds $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOP* PIOP *is* compiler-safe *if for any $\mathtt{i}$ and $\mathtt{x}$, for any $\vec{\pi}$ and any $\vec{\rho}$ there are no polynomial oracles in the last message of the prover and there is an ordering of the focal checks $(\hat{\mathtt{x}}_k)_{k\in\mathcal{K}_f}$ such that (1) for any $k \in \mathcal{K}_f$ we have $\{\hat{\mathtt{x}}_k.a_i : \hat{\mathtt{x}}_k.b_i \notin \mathcal{J}_{k-1}\}$ are ν-independent and (2) we have $\mathcal{J}_{n_e}$ is the set of all the polynomials including index polynomials sent by the prover, where:*

- *$\mathcal{J}_0$ is the set of $n(0)$ index polynomials*
- *for all $k \notin \mathcal{K}_f$, $\mathcal{J}_k \coloneqq \mathcal{J}_{k-1}$*
- *for all $k \in \mathcal{K}_f$, $\mathcal{J}_k \coloneqq \mathcal{J}_{k-1} \cup \{\hat{\mathtt{x}}_k.c_j : j \in [m]\} \cup \{\hat{\mathtt{x}}_k.b_i : i \in [n]\}$*
- *ν is the maximum number of distinct points for which the verifier evaluates a non-index polynomial, i.e.,*
$$\nu \coloneqq \max_{i>n(0)} |\{\hat{\mathtt{x}}_k : [\![p_i]\!] \in \mathsf{oracles}(\mathtt{x}_k), k \in [n_e]\}|$$

Moreover, an $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOP with a delegation phase $\mathsf{PIOP}_1 \| \mathsf{PIOP}_2$ *is* structured *(resp.* compiler-safe*) if* PIOP_1 *and* PIOP_2 *are both structured (resp.* compiler-safe*).*

4.6.2 Polynomial $\hat{\mathcal{R}}_{\mathtt{poly}}$-PIOP

As mentioned in Section 4.1, when designing a new scheme, it is easier to describe the PIOP by specifying a list of polynomial equations between the polynomial oracles as, for example, in [GWC19, CFF+21, RZ21, FFK+23]. In this section we formalize this class of PIOPs using the oracle relation $\hat{\mathcal{R}}_{\mathtt{poly}}$ that upon the instance $\mathtt{x}_{\mathtt{poly}} \coloneqq ((\![p_j]\!])_{j\in[n]}, F, (v_j)_{j\in[n]})$, outputs 1 if and only if:

$$F(p_1(v_1(X)), \ldots, p_n(v_n(X)), X) \equiv 0$$

where $v_j \in \mathbb{F}[X], \forall j$ and $F \in \mathbb{F}[X_1, \ldots, X_n, X]$.

We consider $\hat{\mathcal{R}}_{\mathtt{poly}}$-PIOPs that are *structured* as described below.

Definition 4.6.7 (Structured $\hat{\mathcal{R}}_{\mathtt{poly}}$-PIOP)**.** *An r-rounds $\hat{\mathcal{R}}_{\mathtt{poly}}$-PIOP* PIOP *is* structured *if there exists a deterministic PT algorithm $\tilde{\mathcal{V}}$ such that for any $\mathtt{i}$ and $\mathtt{x}$, for any $\vec{\pi}$ and for any $\vec{\rho}$, $j \in [n]$,$k \in [n_e]$ we have that:*

$$\tilde{v}_{j,k}((\rho_i)_{i\in[r-2]}, X) = \mathtt{x}_k.v_j(X)$$

where $(\tilde{v}_{j,k})_{j,k} \leftarrow \tilde{\mathcal{V}}(\mathbb{F}, \mathtt{i}, \mathtt{x})$ and $\{\hat{\mathtt{x}}_k\}_k \leftarrow \mathcal{V}^{\mathsf{I}(\mathbb{F},\mathtt{i})}(\mathbb{F}, \mathtt{x}, \vec{\pi}, \vec{\rho})$.

Moreover, an $\hat{\mathcal{R}}_{\mathtt{poly}}$-PIOP with a delegation phase $\mathsf{PIOP}_1 \| \mathsf{PIOP}_2$ *is structured if* PIOP_1 *and* PIOP_2 *are both structured.*

We extend the *compiler-safe* definition of [FFK+23] to capture $\hat{\mathcal{R}}_{\mathtt{poly}}$-PIOPs with delegation phase. We require that for each polynomial sent by the prover in the first $r-1$ rounds there must be an equation that involves evaluating it on a (non-constant function of) the last random coin sent by the verifier. Crucially, we require that the prover does not send any polynomial in the last round. We do not make any restriction on the index polynomials. Our notion of compiler-safe is more inclusive than in [FFK+23], as it holds for PIOPs such like Marlin [CHM+20] and Lunar [CFF+21] without any changes.

Definition 4.6.8 (Compiler-safe $\hat{\mathcal{R}}_{\mathtt{poly}}$-PIOP)**.** *An r-rounds $\hat{\mathcal{R}}_{\mathtt{poly}}$-PIOP* PIOP *is* compiler-safe *if for any* $\mathtt{i}$ *and any* $\mathtt{x}$, $\forall \vec{\pi}$, $n(r) = 0$ *and:*

$$\forall j \in [n] \setminus [n(0)] : \exists k \text{ s.t. } \mathsf{deg}_{X_{r-1}}(\tilde{v}_{j,k}) \geq 1$$

where $(\tilde{v}_{j,k})_{j,k} \leftarrow \tilde{\mathcal{V}}(\mathbb{F}, \mathtt{i}, \mathtt{x})$.

Moreover, an $\hat{\mathcal{R}}_{\mathtt{poly}}$-PIOP with a delegation $\mathsf{PIOP}_1 \| \mathsf{PIOP}_2$ *is compiler-safe if* PIOP_1 *and* PIOP_2 *are both compiler-safe.*

4.6.3 From $\hat{\mathcal{R}}_{\mathsf{poly}}$-PIOP to $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOP

Arguably, the notion of compiler safe for $\hat{\mathcal{R}}_{\mathtt{poly}}$-PIOP is very easy to check. In particular, it is much easier to check than the same notion for $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOP. In this section, we show that we can transform a compiler-safe $\hat{\mathcal{R}}_{\mathtt{poly}}$-PIOP into a compiler-safe $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOP. We believe this can give an easy-to-follow recipe when designing new KZG-based SE zkSNARKs from PIOPs because the cryptographer needs only to focus on designing compiler-safe $\hat{\mathcal{R}}_{\mathtt{poly}}$-PIOP.

Our transformation is similar to the general strategy proposed by [GWC19], but it explicitly makes sure that the derived checks result into a compiler-safe $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOP. We start by giving the transform for the case in which there is one[14] polynomial check G and all the $(v_i)_{i\in[n]}$ are equal to some non-constant polynomial v. Let $\mathcal{J}_0$ be the set of the index polynomials, we need to find a subset $\mathcal{I} \subset [n]$ of minimal size $m < n$, and polynomials $(G_i)_{i\in\bar{\mathcal{I}}}$, where $\bar{\mathcal{I}} = [n] \setminus \mathcal{I}$, such that:

- $F(X_1, \ldots, X_n, X) = \sum_{i\in\bar{\mathcal{I}}} G_i((X_j)_{j\in\mathcal{I}}, X) \cdot X_i$,
- $(G_i((p_j(v(X)))_{j\in\mathcal{I}}, X))_{i\notin\mathcal{J}_0}$ are 1-independent.

We define the instance $\hat{\mathtt{x}}_{\mathsf{lin}} := ((\![p_i]\!])_{i\in\mathcal{I}}, (\![p_i]\!])_{i\in\bar{\mathcal{I}}}, (G_i)_{i\in\bar{\mathcal{I}}}, v(\vec{\rho}), 0)$. This transform minimizes the number of *core* polynomial oracles, which results in minimizing the size of the proof of $\mathsf{CP}_{\mathsf{lin}}$.

When there are distinct polynomials v_i the optimization problem gets more complex. In this case, assume that the number of distinct polynomials v_i is equal to ν, then the transformation needs to find sets $\mathcal{I}_1$ and $\mathcal{I}_2$ and minimizes the size of $\mathcal{I} = \mathcal{I}_1 \cup \mathcal{I}_2$, such that it can decompose F into $\nu - 1$ instances for batch-evaluation[15] that check $p_i(v_i(\vec{\rho})) = y_i$ for all $i \in \mathcal{I}_1$ and for values y_i that are sent as part of the last message of the $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOP prover, and one polynomial $F'((X_i)_{i\in\bar{\mathcal{I}}_1}, X) = F((y_i)_{i\in\mathcal{I}_1}, (X_i)_{i\in\bar{\mathcal{I}}_1}, X)$ where, similar to the previous case, F' can be decomposed as:

- $F'((X_i)_{i\in\bar{\mathcal{I}}_1}, X) = \sum_{i\in\mathcal{I}_2} G_i((X_j)_{j\in\bar{\mathcal{I}}}, X) \cdot X_i$,
- $(G_i((p_j(v_j(X)))_{j\in\bar{\mathcal{I}}}, X))_{i\notin\mathcal{J}_0\cup\mathcal{I}}$ are ν-independent.

Finally, when the PIOP has a delegation phase, we can just apply the transform both to PIOP_1 and PIOP_2, this works because the checks involve two disjoint sets of polynomial oracles (excluding the index polynomials, that however are shared among the two phases).

[14] When there are multiple checks with the same non-constant v we can simply batch together the equations in one single equation.

[15] We notice that a $\hat{\mathcal{R}}_{\mathsf{lin}}$-instance can be trivially reduced to a batch-evaluation by having an empty set of right polynomial oracles.

4.6.4 Notable PIOPs

In what follows, we show how to express in $\hat{\mathcal{R}}_{\mathsf{lin}}$ form the PIOPs underlying PLONK and Marlin (with all optimizations); it is easy to extend this analysis to Lunar and Basilisk that are very similar to Marlin and PLONK, respectively.

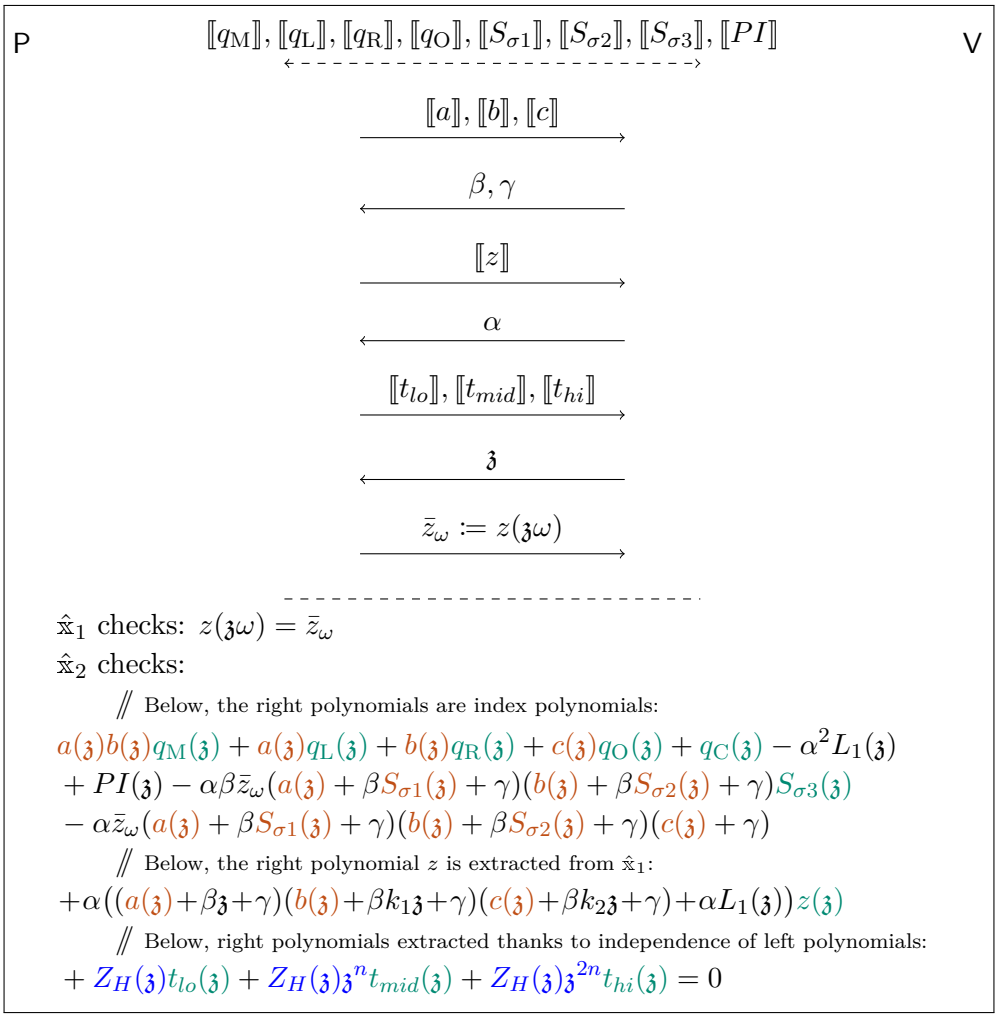


Figure 4.2: The $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOP PLONK. For $\hat{\mathfrak{x}}_2$, we highlight the *core* polynomials, the *left* polynomials that are linearly independent, and the *right* polynomials.

PLONK. We show in Fig. 4.2 how PLONK [GWC19] can be written as a 4-rounds $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOP in which the verifier outputs two checks. The maximum number of distinct points for which the verifier evaluates a non-index polynomial is 2 since the oracle polynomial $[\![z]\!]$ is evaluated on $\mathfrak{z}$ and $\mathfrak{z}\omega$. In $\hat{\mathfrak{x}}_1$ the verifier tests that $z(\mathfrak{z}\omega)$ equals the field element $\bar{z}_\omega$ sent in the last round by the prover. Moreover, all but $[\![t_{lo}]\!], [\![t_{mid}]\!], [\![t_{hi}]\!]$ of the *right* oracle polynomials of $\hat{\mathfrak{x}}_2$ are part of the index or are extracted from $\hat{\mathfrak{x}}_1$. However, the corresponding *left* oracle polynomials, that we highlight in the figure, are linearly independent w.r.t. $\mathbb{F}_{\leq 2}[X]$, which results in a compiler-safe PIOP according to Definition 4.6.6.

Marlin. We show in Fig. 4.3 how Marlin [CHM+20, ark21] can be written as a 3-rounds $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOP with a Delegation Phase.

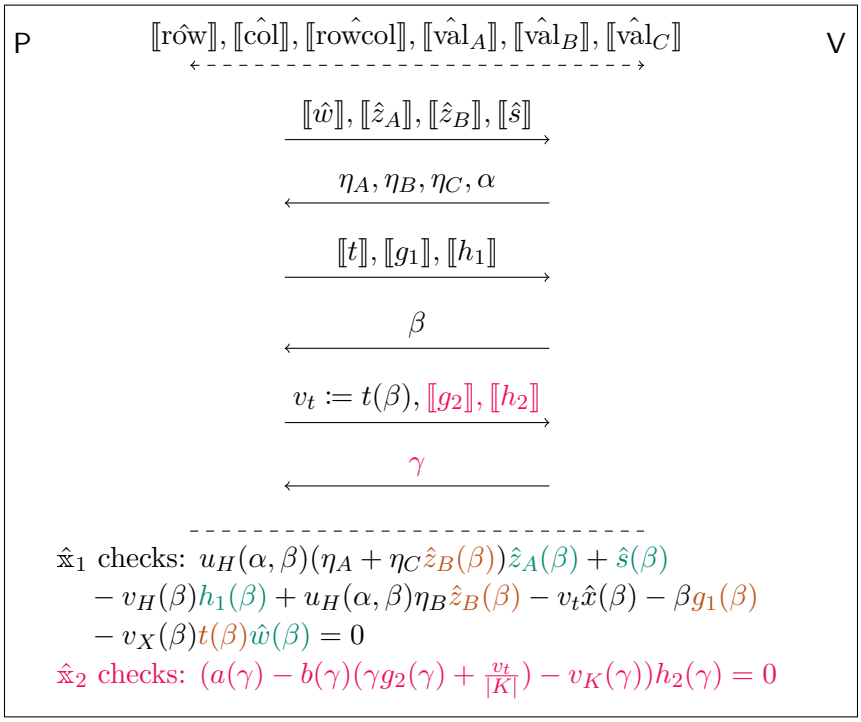


Figure 4.3: The $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOP Marlin, where: v_H (resp. v_K, v_X) denotes the vanishing polynomial of the subgroup H (resp. K, X) of $\mathbb{F}$; u_H is the formal derivative of v_H; the polynomials a and b are computed using the index polynomials and the coins α and β. We highlight the delegation phase, the *core* and *right* polynomials of $\hat{\mathfrak{x}}_1$.

4.7 Revisiting the PIOP-to-zkSNARK compiler

We show how to turn compiler-safe $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOPs into simulation-extractable zkSNARKs. We stress that, although the formalism we adopt differs from previous work, the resulting compiler's construction is the usual one with the linearization trick optimization.

Definition 4.7.1. *We say that* Π *is* strong simulation-extractable in the algebraic group model with oblivious sampling *if and only if* Π *is* $\mathbf{\Phi}_{\mathsf{sse}}$*-simulation-extractable where for any* (Φ_0, Φ_1) *in the family of policies* $\mathbf{\Phi}_{\mathsf{sse}}$ *we have that* Φ_0 *outputs group elements* $\mathsf{coms} = (\mathsf{c}_i)_i$ *from an Aff-MDH secure and witness sampleable distribution and* Φ_1 *checks that the forgery* $(\mathbb{x}^*, \pi^*) \notin \mathcal{Q}_{\mathsf{sim}}$.

Theorem 4.7.1. *Let* $\mathsf{CP}_{\mathsf{lin}}$ *be the CP-SNARK for* $\mathcal{R}_{\mathsf{lin}}$ *defined in Section 4.5.2. Let* PIOP *be a compiler-safe* $\hat{\mathcal{R}}_{\mathsf{lin}}$*-PIOP for relation* $\mathcal{R}$ *that is state-restoration straight-line extractable, bounded zero-knowledge, and with simulation-friendly polynomial oracles. Let* Π *be the zk-SNARK compiled from* PIOP *using the compiler in Fig. 4.4. Then* Π *is zero-knowledge and strong simulation-extractable in the AGM. Furthermore, if* $\mathcal{R}$ *is an oracle relation, then* Π *is a CP-SNARK.*

Proof intuition. The proof of zero-knowledge follows rather easily from the bounded zero-knowledge, the simulation-friendly polynomial oracles and the (leaky) zero-knowledge of $\mathsf{CP}_{\mathsf{lin}}$. Moreover, the simulation strategy makes sure that the view of the adversary is always algebraic

$\Pi.\mathsf{Derive}(\mathsf{srs}, \mathbb{i})$:

$\vec{p}_0 \leftarrow \mathsf{I}(\mathbb{F}, \mathbb{i})$;
for $j \in [n(0)]$ **do** : $\mathsf{c}_j \leftarrow [p_{0,j}(s)]_1$
$\mathsf{ek}_\mathbb{i} \leftarrow \vec{p}_0,\ \mathsf{vk}_\mathbb{i} \leftarrow (\mathsf{c}_j)_{i\in[n(0)]}$
return $(\mathsf{ek}_\mathbb{i}, \mathsf{vk}_\mathbb{i})$

$\Pi.\mathsf{Verify}(\mathsf{vk}_\mathbb{i}, \mathbb{x}, \pi_\Pi)$:

derive $(\bar{\pi}_1, \ldots, \bar{\pi}_r)$
for $i \in [r(|\mathbb{i}|) - 1]$ **do** : // Fiat-Shamir transform
  $\rho_i \leftarrow \mathsf{RO}(\mathsf{vk}_\mathbb{i}, \mathbb{x}, \bar{\pi}_1, \ldots, \bar{\pi}_i)$
$\{\hat{\mathbb{x}}_k\}_k \leftarrow \mathcal{V}(\mathbb{F}, \mathbb{x}, \vec{\pi}, \vec{\rho})$
return $\bigwedge_{k\in[n_e]} \mathsf{Verify}_{\mathsf{lin}}(\mathsf{srs}, \mathbb{x}_k, \pi_k)$

$\Pi.\mathsf{Prove}(\mathsf{srs}, \mathsf{ek}_\mathbb{i}, \mathbb{x}, \mathbb{w})$:

for $i \in [r(|\mathbb{i}|)]$ **do** :
  // Get messages from PIOP prover
  $(\vec{p}_i, \vec{\pi}_i) \leftarrow \mathsf{P}(\mathbb{F}, \mathbb{i}, \mathbb{x}, \mathbb{w}, \rho_1, \ldots, \rho_{i-1})$
  $t \leftarrow \sum_{j\in[i-1]} n(j)$
  for $j \in [n(i)]$ **do** : $\mathsf{c}_{t+j} \leftarrow [p_{i,j}(s)]_1$
  // Fiat-Shamir commitments and messages of this round
  $\bar{\pi}_i := (\mathsf{c}_{t+1}, \ldots, \mathsf{c}_{t+n(i)}, \vec{\pi}_i)$
  if $i < r$: $\rho_i \leftarrow \mathsf{RO}(\mathsf{vk}_\mathbb{i}, \mathbb{x}, \bar{\pi}_1, \ldots, \bar{\pi}_i)$
$\{\hat{\mathbb{x}}_k\}_k \leftarrow \mathcal{V}^{\mathsf{I}(\mathbb{F},\mathbb{i})}(\mathbb{F}, \mathbb{x}, \vec{\pi}, \vec{\rho})$
for $k \in [n_e] : \pi_k \leftarrow \mathsf{Prove}_{\mathsf{lin}}(\mathsf{srs}, \mathbb{x}_k, (p_i : i \in \mathsf{indexes}(\hat{\mathbb{x}}_k)))$
return $(\vec{\mathsf{c}}, \vec{\pi}, (\pi_k)_k)$

Figure 4.4: The compiler based from $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOPs and KZG commitment scheme to Universal zkSNARKs. We associate to the instance $\hat{\mathbb{x}}_k$ for the oracle relation $\hat{\mathcal{R}}_{\mathsf{lin}}$ the instance $\mathbb{x}_k$ for the commit-and-prove relation $\mathcal{R}_{\mathsf{lin}}$, the latter instance is identical to $\hat{\mathbb{x}}_k$ but where any oracle $[\![p]\!] \in \mathsf{oracles}(\hat{\mathbb{x}}_k)$ is substituted with the commitment $[p(s)]_1$.

consistent and that the maximum nesting level is at most equal to the number of $\hat{\mathcal{R}}_{\mathsf{lin}}$ instances queried by the verifier in a single proof.

For the proof of simulation extractability, we need to show that for any adversary $\mathcal{A}$, there is an extractor $\mathcal{E}$ who outputs a valid witness whenever $\mathcal{A}$ submits a valid forgery, i.e., a new and valid pair of statement and proof. To do that, we reduce the simulation extractability of Π to the state-restoration knowledge soundness of the PIOP. Our extractor parses the algebraic representation of the commitments sent by the adversary when querying the RO (to compute the next coin of the verifier), and extracts from them the underlying polynomials. The latter polynomials are used as the prover oracles sent in the reduction to the state-restoration knowledge soundness experiment to define a *verifier state* and retrieve the next coins of the verifier. Notice that this reduction would not work if the adversary used simulated elements in (the transcript) of his forgery. We bound the probability of this bad event by reducing to the simulation extractability of $\mathsf{CP}_{\mathsf{lin}}$. More in detail, for each focal check we perform one reduction to the Φ_{lin}-SE of $\mathsf{CP}_{\mathsf{lin}}$ and in the k-th reduction we extract the polynomials $\mathcal{J}_k$ with the knowledge of the polynomials in $\mathcal{J}_{k-1}$ (i.e., we use partial extractability), where the sets $(\mathcal{J}_k)_k$ come from the compiler-safeness (c.f. Definition 4.6.6).

Proof. In what follows, we assume that PIOP is in fact an $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOP with a delegation phase, i.e., the prover interacts with the verifier for $r + 1$ rounds.

Zero-Knowledge. We start showing the zero-knowledge simulator for Π and an adversary submitting Q queries. The simulator is in Fig. 4.5. We define the bounded (leakage) list $\mathsf{Leak}_{\hat{\mathbb{x}}}$ as the list of all tuples $(i, \hat{\mathbb{x}}.x)$ such that $i \in \mathsf{indexes}(\hat{\mathbb{x}})$ and $[\![p_i]\!]$ is a left oracle polynomial of $\hat{\mathbb{x}}$. We recall that $\mathsf{indexes}(\hat{\mathbb{x}})$ is the list of indexes of the polynomial oracles sent by the prover that are involved in the instance $\hat{\mathbb{x}}$.

The zero-knowledge guarantees of the simulator come from the simulation-friendly polynomial oracles of PIOP (see Definition 4.6.3), the bounded zero-knowledge property of the PIOP, and the L_{lin}-leaky zero-knowledge property of $\mathsf{CP}_{\mathsf{lin}}$, where $L_{\mathsf{lin}}(\mathbb{x}, \mathbb{w}) \coloneqq (\mathbb{w}.C_j(\mathbb{x}.x))_j$ (cf. Section 4.5.2).

We can show this with a simple hybrid argument. Let $\mathbf{G}_0$ be the *real-world* experiment where $\mathcal{A}$ interacts with a real prover and the random oracle.

- The first hybrid $\mathbf{G}_1$ is the same as $\mathbf{G}_0$ but where, at every call to the prover, we additionally compute the bounded (leakage) list Leak computed by the real prover. This hybrid is obliviously identically distributed to the previous one.

- The second hybrid $\mathbf{G}_2$ is the same as $\mathbf{G}_1$ but where the SRS is generated using $\mathsf{CP}_{\mathsf{lin}}.\mathcal{S}_0$ and the proofs of $\mathcal{R}_{\mathsf{lin}}$ for the instances in $\mathcal{K}_1$ are generated using the simulator $\mathsf{CP}_{\mathsf{lin}}.\mathcal{S}_1$. Notice that, since $\mathsf{CP}_{\mathsf{lin}}$ is only leaky-zero-knowledge, to generate such proofs, the simulator additionally needs the leakage which we can compute using the bounded list $\mathsf{Leak}_{\hat{\mathbb{x}}}$ over the polynomials computed by the prover. The proof of indistinguishability between the two hybrids follows easily from the leaky zero-knowledge property of $\mathsf{CP}_{\mathsf{lin}}$.

- The third hybrid $\mathbf{G}_3$ is the same as $\mathbf{G}_2$ but where, at every call to the prover, we sample the commitments as uniformly random $\mathbb{G}_1$-group elements. For this step we use that PIOP (or PIOP_1, if $\mathsf{PIOP} = \mathsf{PIOP}_1 \| \mathsf{PIOP}_2$ has a delegation phase) has simulation-friendly polynomial oracles (cf. Definition 4.6.3).

- The last hybrid $\mathbf{G}_4$ is the same a $\mathbf{G}_3$ but where (1) we finally switch to use the zero-knowledge of the $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOP and (2) we use the leakage using the simulator. In particular, we need to use the simulator for an (honest) verifier that samples its messages by computing the random oracle on the transcript so far[16]. This allows showing that the simulator is in the non-programmable random oracle. The last hybrid is identically distributed to the *ideal-world* experiment where $\mathcal{A}$ interacts with the simulator and the random oracle.

Finally, we show that, independently of the strategy of the adversary $\mathcal{A}$, the view at the end of the simulation extractability experiment for the compiled zkSNARK and with the simulator described in Fig. 4.5 is algebraic consistent.

When the relation $\mathcal{R}$ is not an oracle relation, namely when the instances do not contain any commitment, we can easily show that the view is algebraic consistent by noticing that the simulator, at the q-th query, produces n_e simulated proofs on the commitments $\mathsf{coms}^{(q)}$, moreover, since the PIOP prover can only prove algebraic consistent statements then also the simulator can only compute algebraic consistent statements, otherwise we would have a distinguisher for the zero-knowledge of the PIOP.

We need a more careful analysis when the relation $\mathcal{R}$ is an oracle relation, in which case Π is a CP-SNARK. In fact, assume that, at the q-th simulation query, the adversary includes a simulated commitment $\tilde{\mathsf{c}}$ in the queried instance $\mathbb{x}$, namely either $\tilde{\mathsf{c}} \in \mathsf{coms}^{(j)}$ for $j < q$ or $\tilde{\mathsf{c}} \in \mathsf{coms}'$. We observe that the simulator trivially preserves the algebraic consistency across multiple proofs for *focal* checks since they involve evaluation on random coins, thus on evaluation points that were not queried before. As for non-focal checks, notice that, because of

[16]Technically, we can hardcode the full description of the random oracle inside the verifier and rely on the statistical zero-knowledge property.

$\mathcal{S}(0, \mathsf{pp}_{\mathbb{G}})$

$\mathsf{srs}, \mathsf{st}_{\mathsf{CP}} \leftarrow\!\$\ \mathsf{CP}_{\mathsf{lin}}.\mathcal{S}(0, \mathsf{pp}_{\mathbb{G}})$
$\mu \leftarrow 0 \quad /\!/\ \mathcal{S}_1$ queries counter
for $j \in [Q]$ **do** :
  $\mathsf{coms}^{(j)} \leftarrow\!\$\ \mathbb{G}_1^{n-n(r)}$
$\mathsf{coms}' \leftarrow\!\$\ \Phi_0(\mathsf{pp}_{\mathbb{G}})$
$\mathsf{st}_{\mathcal{S}} \leftarrow (\mathsf{st}_{\mathsf{CP}}, \mu, (\mathsf{coms}^{(j)})_j, \mathsf{coms}')$
return $\mathsf{srs}, \mathsf{st}_{\mathcal{S}}$

$\mathcal{S}(2, \mathsf{st}_{\mathcal{S}}, s, \mathsf{aux})$

if $(s, \mathsf{aux}, a) \in \mathcal{Q}_{\mathsf{RO}}$:
  return $a, \mathsf{st}_{\mathcal{S}}$
$a \leftarrow\!\$\ \mathbb{F}$
$\mathcal{Q}_{\mathsf{RO}} \leftarrow \mathcal{Q}_{\mathsf{RO}} \cup (s, \mathsf{aux}, a)$
return $a, \mathsf{st}_{\mathcal{S}}$

$\mathcal{S}(1, \mathsf{st}_{\mathcal{S}}, \mathsf{srs}, (\mathbb{i}, \mathbb{x}))$

$\mathsf{st}_{\mathcal{S}} \leftarrow (\mathsf{st}_{\mathsf{CP}}, \mu, (\mathsf{coms}^{(j)})_j, \mathsf{coms}')$
$\mathsf{c}_1, \ldots \mathsf{c}_{n-n(r)} \leftarrow \mathsf{coms}^{(\mu)}$
for $i \in [r-1]$ **do** :
  $\vec{\pi}_i \leftarrow \mathsf{PIOP}.\mathcal{S}_0(\mathbb{F}, \mathbb{i}, \mathbb{x}, \rho_1, \ldots, \rho_{i-1})$
  $t \leftarrow \sum_{j \in [i-1]} n(j)$
  $\bar{\pi}_i = (\mathsf{c}_t, \ldots, \mathsf{c}_{t+n(i)}, \vec{\pi}_i)$
  $\rho_i \leftarrow \mathsf{RO}(\mathsf{vk}_{\mathbb{i}}, \mathbb{x}, \bar{\pi}_1, \ldots, \bar{\pi}_i)$
$(\vec{p}_r, \vec{\pi}_r) \leftarrow \mathsf{P}_2(\mathbb{F}, \mathbb{i}, \mathbb{x}, \vec{\pi}, \rho_1, \ldots, \rho_{r-1}) \| \mathsf{PIOP}.\mathcal{S}_0(\mathbb{F}, \mathbb{i}, \mathbb{x}, \rho_1, \ldots, \rho_{r-1})$
$(\mathsf{c}_{n-n(r)+1}, \ldots, \mathsf{c}_n) \leftarrow ([p_{r,j}(s)]_1)_{j \in [n(r)]}$
$\bar{\pi}_r = (\mathsf{c}_{n-n(r)+1}, \ldots, \mathsf{c}_n, \vec{\pi}_r)$
$\rho_r \leftarrow \mathsf{RO}(\mathsf{vk}_{\mathbb{i}}, \mathbb{x}, \bar{\pi}_1, \ldots, \bar{\pi}_r)$
$\vec{\pi}_{r+1} \leftarrow \mathsf{P}_2(\mathbb{F}, \mathbb{i}, \mathbb{x}, \vec{\pi}, \vec{\rho})$
$\{\hat{\mathbb{x}}_k\}_{k \in \mathcal{K}_1}, \{\hat{\mathbb{x}}_k\}_{k \in \mathcal{K}_2} \leftarrow \mathcal{V}^{\mathsf{I}(\mathbb{F}, \mathbb{i})}(\mathbb{F}, \mathbb{x}, \vec{\pi}, \vec{\rho})$
for $k \in \mathcal{K}_1$:
  $\mathsf{leak} \leftarrow \mathsf{PIOP}.\mathcal{S}_1(\mathbb{F}, \mathbb{i}, \hat{\mathbb{x}}_k, \mathsf{Leak}_{\hat{\mathbb{x}}_k})$
  $\pi_k \leftarrow \mathsf{CP}_{\mathsf{lin}}.\mathcal{S}_1(\mathsf{st}_{\mathsf{CP}}, \mathbb{x}_k, \mathsf{leak})$
for $k \in \mathcal{K}_2$:
  $\pi_k \leftarrow \mathsf{CP.Prove}(\mathsf{srs}, \mathbb{x}_k, (p_i : i \in \mathsf{indexes}(\hat{\mathbb{x}}_k)))$
$\pi \leftarrow (\vec{\mathsf{c}}, \vec{\pi}, (\pi_k)_k)$
$\mathsf{st}_{\mathcal{S}} \leftarrow (\mathsf{st}_{\mathsf{CP}}, \mu + 1, (\mathsf{coms}^{(j)})_j, \mathsf{coms}')$
return $\pi, \mathsf{st}_{\mathcal{S}}$

Figure 4.5: The simulator $\mathcal{S}$ for a Π compiled from an r-rounds $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOP with a delegation phase. We highlight the parts needed only for the delegation setting.

the $\vec{b}$-bounded zero-knowledge of the PIOP we have that $\tilde{\mathsf{c}}$ (or better say the oracle associated to it) must be a right polynomial oracle in all the $\hat{\mathcal{R}}_{\mathsf{lin}}$ instances where it appears. Otherwise, the PIOP would not be able to support leakage on this oracle. Moreover, in all the non-focal instances where it appears as a left polynomial either the instance contains another left polynomial that is sent by the prover or the instance evaluates to a constant value y, again, because of the bounded zero-knowledge property of the PIOP. Notice, in the first case the algebraic consistency holds because $\tilde{\mathsf{c}}$ is evaluated together with a simulated commitment from $\mathsf{coms}^{(q)}$. In the second case, algebraic consistency holds because $\tilde{\mathsf{c}}$ is evaluated on an evaluation point and to a constant value, thus consistently with the previous simulated proofs.

Simulation Extractability. We define the extractor for Π for a given adversary $\mathcal{A}_\Pi$. We make some simplifying assumptions on the behavior of $\mathcal{A}_\Pi$: (1) the adversary always queries first the RO on a string that can be parsed as $(\mathbb{i}, \mathbb{x})$ before querying the simulation oracle on the same string, (2) the auxiliary string $\mathsf{aux}_{\mathcal{E}}$ output by $\mathcal{A}_\Pi$ can be parsed as a list of strings $(s_i, \mathsf{aux}_i, \mathsf{st}_i)_i$ and a string $\mathsf{aux}'_{\mathcal{E}}$ where for any i we have $(s_i, \mathsf{aux}_i, \mathsf{st}_i)$ string is identical to the auxiliary input output at the i-th query of the adversary. These assumptions are without loss of generality. In fact, given an adversary $\mathcal{A}_\Pi$ that does not respect these rules we can always define another

adversary that runs internally $\mathcal{A}_{\Pi}$, collects all the necessary information to comply with (2) and moreover follows the rule (1).

Finally, we emphasize that in our proof strategy we need to extract the witness polynomials from the proofs according to some order, thus we would first need to sort the proofs of the adversary and then proceed. However, w.l.o.g., we assume that the focal checks output by the verifier are already sorted so to match the compilation-safeness property (Definition 4.6.6), and such that all the "delegation checks", namely $(\hat{\mathbb{x}}_k)_{k\in\mathcal{K}_2}$, come before the others: notice that it is always possible to define such sorting since the non-delegation checks and the delegation checks involve two disjoint sets of polynomials (excluding the index polynomials that however are already in $\mathcal{J}_0$).

Before proceeding, we set some notation:

- Let $\mathsf{oracles}_b(\hat{\mathbb{x}})$ and $\mathsf{oracles}_c(\hat{\mathbb{x}})$ be respectively the right and the core oracles of $\hat{\mathbb{x}}$. Similarly, let $\mathsf{indexes}_b(\hat{\mathbb{x}})$ and $\mathsf{indexes}_c(\hat{\mathbb{x}})$ be respectively the indexes of the right and the core oracles of $\hat{\mathbb{x}}$.

- For all k, let $\mathcal{CI}(\hat{\mathbb{x}}_k) := \mathsf{oracles}_b(\hat{\mathbb{x}}_k) \cap \mathcal{J}_k$ be the *compiler-safe index set*, namely the set of the indexes of the polynomials sent by the prover that are either part of the index or may be extracted from $\hat{\mathbb{x}}_{k'}$, for $k' < k$.

- For all k, we define $\{\gamma_{j,k}\}_j := \mathsf{indexes}_c(\hat{\mathbb{x}}_k)$ and $\{\beta_{i,k}\}_i := \mathsf{indexes}_b(\hat{\mathbb{x}}_k)$. Whenever it is clear from the context, we may omit the index k.

- Let $\mathcal{P}_i$ be the indexes of the polynomials sent at the i-th round by the prover.

- Given a proof π for Π, we define *the RO-queries of* π the list of strings

$$((\mathsf{vk}_{\mathsf{i}}, \mathbb{x}), \ldots, (\mathsf{vk}_{\mathsf{i}}, \mathbb{x}, \bar{\pi}_1, \ldots, \bar{\pi}_r))$$

- We say that the adversary *copied up to round* i (the transcript of) its proof π_{Π} from a simulated proof π'_{Π} if the first $i+1$ entries of their RO-queries are equal.

- We say that a proof π_{Π} *uses a simulated element* if there is a non-zero coefficient depending on the simulated elements provided by $\mathcal{S}$ in the algebraic representation of any of the commitments in π_{Π}.

- We say that a coin is *fresh* if it does not appear in any of the simulated transcripts of the proofs output by $\mathcal{S}$.

- We use $\mathcal{K}_1$ and $\mathcal{K}_2$ to denote the indexes of the checks output in PIOP_1 and PIOP_2 respectively, where $\mathsf{PIOP} := \mathsf{PIOP}_1 \| \mathsf{PIOP}_2$

- We let $\mathcal{E}_{\mathsf{Com}}$ be the canonical AGM extractor of the KZG polynomial commitment, namely the one that parses the algebraic representation of a commitment c as $(f, \vec{r})$ and returns the polynomial $f(X)$. Notice the extractor fails when $\vec{r} \neq \vec{0}$.

The extractor $\mathcal{E}_{\Pi}(\mathbb{x}_{\Pi}, \pi_{\Pi}, \mathsf{aux}_{\mathcal{E}})$:

1. Parse $\mathsf{aux}_{\mathcal{E}}$ as the concatenation of a list $(s_i, \mathsf{aux}_i, \mathsf{st}_i)_i$ and $\mathsf{aux}'_{\mathcal{E}}$, where $(s_i, \mathsf{aux}_i, \mathsf{st}_i)$ is the output of $\mathcal{A}_{\Pi}$ at the i-th query to the ROM and $\mathsf{aux}'_{\mathcal{E}}$ the remaining auxiliary information given by the adversary (namely, the auxiliary information associated with its last output).
2. Take the commitments $(\mathsf{c}_i)_{i\in[n]}$ in both $\mathbb{x}_{\Pi}$ (if it is a commit-and-prove relation) and π_{Π}; from π_{Π} derive the messages $\bar{\pi}_1, \ldots, \bar{\pi}_r$ and find the indexes q_i such that $s_{q_i} = (\mathsf{vk}_{\mathbb{i}}, \mathbb{x}, \bar{\pi}_1, \ldots, \bar{\pi}_i)$.
3. Return $\perp$ if π_{Π} uses a simulated element of one of the proofs.
4. For $i \in [n]$, let $p_i \leftarrow \mathcal{E}_{\mathsf{Com}}(\mathsf{c}_i)$.
5. Return $\perp$ if for some i, $\exists j \in \mathcal{P}_i : \mathsf{c}_j \neq [p_i(s)]_1$.
6. Let $\hat{\mathbb{x}}_k = ((\mathsf{c}_{\gamma_j})_{j\in[m]}, (\mathsf{c}_{\beta_i})_{i\in[n]}, (G_i)_{i\in[n]}, x, y)$. If $\exists k$: $\sum_i G_i(p_{\gamma_1}(x), \ldots, p_{\gamma_m}(x), x)p_{\beta_i}(x) \neq y$, return $\perp$.
7. Return $\mathcal{E}_{\mathsf{PIOP}}(\mathbb{i}, \mathbb{x}_{\Pi}, (p_j)_j)$

To analyze the success of the extractor we define a series of hybrid games. We start from the first hybrid that is the $\mathbf{Exp}^{sr}_{\mathcal{A}_{\mathsf{PIOP}},\mathsf{PIOP}}(\mathbb{F})$ experiment for PIOP (see Definition 3.6.2) for an adversary $\mathcal{A}_{\mathsf{PIOP}}$ that we define next.

The adversary $\mathcal{A}_{\mathsf{PIOP}}$:

1. Run simulator $\mathsf{srs}, \mathsf{st}_{\mathcal{S}} \leftarrow \mathcal{S}(0, \mathsf{pp}_{\mathbb{G}})$ and set $\mathcal{Q}_{\mathsf{RO}}, \mathcal{Q}_{\mathsf{sim}}$ empty sets.
2. Run $\mathcal{A}_{\Pi}(\mathsf{srs})$ and answer all the simulation queries of $\mathcal{A}_{\Pi}$ with $\mathcal{S}_1$.
3. Upon i-th query (s_i, aux_i) from $\mathcal{A}_{\Pi}$ to $\mathcal{S}_2$:
   (a) if s_i is in the RO-queries of a simulated proof in $\mathcal{Q}_{\mathsf{sim}}$ then run $\mathcal{S}_2$ on input s_i.
   (b) Else parse s_i as a (partial) transcript $\mathtt{trns} = (\mathsf{vk}_{\mathbb{i}}, \mathbb{x}, \bar{\pi}_1, \ldots, \bar{\pi}_{r'})$; parse $\bar{\pi}_j$ as $(\vec{\mathsf{c}}_j, \vec{\pi}_j)$; find the witness polynomials $\mathbb{w} := (f_{r',j})_j$ that are the algebraic representation (depending on ck) of $\vec{\mathsf{c}}_{r'}$; find in $\mathsf{SeenStates}$ the state $\mathsf{cvs} = (\mathbb{i}, \mathbb{x}, \vec{\pi}_1, \{p_{1,i}\}_i \| \rho_1 \| \ldots \| \vec{\pi}_{r'-1}, \{p_{r'-1,i}\}_i \| \rho_{r'-1})$, set the verifier state to cvs, and send the message $(\mathbb{w}, \vec{\pi}_{r'})$ to the PIOP verifier. Receive the challenge $\rho_{r'}$ from the verifier. Finally, program the random oracle adding $(s_i, \mathsf{aux}_i, \rho_{r'})$ to $\mathcal{Q}_{\mathsf{RO}}$.
4. Eventually the adversary outputs a valid forgery $(\mathbb{x}_{\Pi}, \pi_{\Pi})$. From π_{Π} derive the (full) PIOP transcript $\mathtt{trns} := (\mathbb{i}, \mathbb{x}, \bar{\pi}_1, \rho_1, \ldots, \bar{\pi}_r)$. Let i be the index of RO query of the partial transcript $(\mathbb{i}, \mathbb{x}, \bar{\pi}_1, \rho_1, \ldots, \bar{\pi}_{r-1})$; as described in the previous step, find the cvs in $\mathsf{SeenStates}$ associated with s_i, set the verifier state to cvs, extract the (last) witness polynomials $\mathbb{w}$ and send $(\mathbb{w}, \vec{\pi}_r)$ to the verifier. The state cvs and the last messages $(\mathbb{w}, \vec{\pi}_r)$ define a full transcript: this would trigger the verifier to perform the decision phase of the PIOP and set the decision bit d of the game.

Let $\mathbf{H}_0$ be the $\mathbf{Exp}^{sr}_{\mathcal{A}_{\mathsf{PIOP}},\mathsf{PIOP}}(\mathbb{F})$. By the state-restoration knowledge extractability of PIOP:

$$\Pr[\mathbf{H}_0] \in \mathsf{negl}(|\mathbb{F}|)$$

Consider $\mathbf{H}_1$ that additionally returns 1 if the adversary $\mathcal{A}_\Pi$ copies a simulated transcript up to and including the last round from a simulated proof π'_Π.

Lemma 4.7.1. $\Pr[\boldsymbol{H}_1] \leq \Pr[\boldsymbol{H}_0] + \epsilon_{\mathsf{lin}}$

Proof. We have that the forged proof π_Π and a simulated proof π'_Π share the same commitments $\vec{\mathsf{c}}$, the same messages $\vec{\pi}$, and therefore the same set of instances $(\hat{\mathbb{x}}_k)_{k\in[n_e]}$.

We parse $\pi_\Pi = (\vec{\mathsf{c}}, \vec{\pi}, (\pi_k)_{k\in[n_e]})$ and the simulated proof as $\pi'_\Pi = (\vec{\mathsf{c}}, \vec{\pi}, (\pi'_k)_{k\in[n_e]})$, if the two hybrids diverge then there must be an index $k \in [n_e]$ such that the proof $\pi_k \neq \pi'_k$, where, depending on the index k,π'_k was either honestly generated (for $k \in \mathcal{K}_1$) or was simulated using $\mathsf{CP}_{\mathsf{lin}}.\mathcal{S}_1$ (for $k \in \mathcal{K}_0$).

Let $\pi_k = (\bar{\pi}, (y_j)_j)$ and $\pi'_k = (\bar{\pi}', (y'_j)_j)$, by case analysis, either $\exists j : y_j \neq y'_j$ or $\forall j : y_j = y'_j$ and $\bar{\pi} \neq \bar{\pi}'$.

The latter case cannot happen by the uniqueness of KZG proofs, notice the uniqueness of KZG proofs holds even when the trapdoor is known by the adversary. The former case is covered by the $\Phi^{\mathcal{J},\nu}_{\mathsf{lin}_+}$-simulation extractability of $\mathsf{CP}_{\mathsf{lin}}$. We describe below a reduction.

Reduction $\mathcal{B}(\mathsf{srs}, \mathsf{pp}_{\Phi_{\mathsf{lin}}})$

1. Run $\mathcal{A}_\Pi(\mathsf{srs})$.
2. Upon query $((\mathbb{i}, \mathbb{x}), \mathsf{aux})$ to the simulation oracle, run the same strategy of $\mathcal{S}_1$ defined in Fig. 4.5, namely using $\mathsf{CP}_{\mathsf{lin}}.\mathcal{S}_1$.
3. Given the forgery $((\mathbb{i}, \mathbb{x}_\Pi), \pi_\Pi)$ output by $\mathcal{A}_\Pi$, find the instance $\hat{\mathbb{x}}_k$ and the corresponding proof π_k; submit the forgery $(\hat{\mathbb{x}}_k, \pi_k)$

We recall that the policy $\Phi^{\mathcal{J},\nu}_{\mathsf{lin}_+}$, since the instance for π_k and π'_k is the same, holds when $\exists j : y_j \neq y'_j$. Thus, by the Theorem 4.5.3, we bound the probability that the $\mathcal{B}$ wins to ϵ_{lin}. □

Consider the hybrid $\mathbf{H}_2$ that, for all $i \in [n]$, computes $p_i \leftarrow \mathcal{E}_{\mathsf{Com}}(\mathsf{c}_i)$ and, additionally, returns 1 if

$$\exists k \in \mathcal{K}_2 : \sum_i G_i(p_{\gamma_1}(x), \dots p_{\gamma_m}(x), x) \cdot p_{\beta_i}(x) \neq y \quad \vee$$
$$\exists j \in \mathcal{P}_r : \mathsf{c}_j \neq [p_j(s)]_1$$

where $\hat{\mathbb{x}}_k = \Big((\mathsf{c}_{\gamma_j})_j, (\mathsf{c}_{\beta_i})_i, (G_i)_i, x, y)\Big)$.

Lemma 4.7.2. $\Pr[\boldsymbol{H}_2] \leq \Pr[\boldsymbol{H}_1] + |\mathcal{K}_2| \cdot \epsilon_{\mathsf{lin}}$

Proof. Notice that, because of the winning condition added in $\mathbf{H}_1$, we can focus on the case in which the adversary $\mathcal{A}_\Pi$ does not copy a simulated transcript (up to and including the last round) from a simulated proof π'_Π. We also notice that, by our simplifying assumption on the order of the verifier's queries, we have $\mathcal{K}_2 = \{1, \dots, |\mathcal{K}_2|\}$, namely, the indexes in $\mathcal{K}_2$ are consecutive numbers.

We prove this lemma through a series of hybrids. Let $\mathbf{H}_{1,0} \equiv \mathbf{H}_1$. For any $k > 1$, let $\mathbf{H}_{1,k}$ be the same as $\mathbf{H}_{1,k}$, but that additionally returns 1 if:

$$\sum_i G_i(p_{\gamma_1}(x), \dots, p_{\gamma_m}(x), x) p_{\beta_i}(x) \neq y \quad \vee$$
$$\exists i : \mathsf{c}_{\beta_i} \neq [p_{\beta_i}(s)]_1 \vee \exists j : \mathsf{c}_{\gamma_j} \neq \Big[p_{\gamma_j}(s)\Big]_1$$

where $\hat{\mathbb{x}}_k = \Big((\mathsf{c}_{\gamma_j})_j, (\mathsf{c}_{\beta_i})_i, (G_i)_i, x, y)\Big)$. Finally, we have that $\mathbf{H}_2 \equiv \mathbf{H}_{1,|\mathcal{K}_2|}$.

Let $\mathcal{I} \coloneqq \mathcal{CI}(\hat{\mathbb{x}}_k)$ be the *compiler-safe index set* of $\hat{\mathbb{x}}_k$ (namely, the set of polynomials that the PIOP's specification guarantees we can extract from the instance $\hat{\mathbb{x}}_k$, see Section 4.7). We reduce to the $(\mathbf{\Phi}^{\mathcal{I},\nu}_{\mathsf{lin}}, \mathcal{F}_{\mathcal{I},\nu})$-simulation extractability of $\mathsf{CP}_{\mathsf{lin}}$. We define the reduction $\mathcal{B}_{\mathsf{lin},k}$.

Reduction $\mathcal{B}_{\mathsf{lin},k}(\mathsf{srs}, \mathsf{pp}_{\Phi_{\mathsf{lin}}})$

1. Run $\mathcal{A}_{\mathsf{\Pi}}(\mathsf{srs})$.
2. Upon query $((\mathbb{i}, \mathbb{x}), \mathsf{aux})$ to the simulation oracle, run the same strategy of $\mathcal{S}_1$ defined in Fig. 4.5 and use oracle access to $\mathsf{CP}_{\mathsf{lin}}.\mathcal{S}_1$.
3. Upon query to $\mathcal{S}_2$:
   (a) If it can be parsed as a partial transcript $\mathtt{trns} = (\mathsf{vk}_{\mathbb{i}}, \mathbb{x}, \bar{\pi}_1, \ldots, \bar{\pi}_r)$, derive the list of single-variable polynomials $(\tilde{v}_{k'})_{k'} \leftarrow \tilde{\mathcal{V}}_2(\mathbb{F}, \mathbb{i}, \mathbb{x})$ and forward the query to $\mathsf{CP}_{\mathsf{lin}}.\mathcal{S}_2$ adding the polynomial $(\tilde{v}_k(X))$ to the auxiliary information.
   (b) Otherwise, simply forward the query to the simulator
4. Given the forgery $((\mathbb{i}, \mathbb{x}_{\mathsf{\Pi}}), \pi_{\mathsf{\Pi}})$ output by $\mathcal{A}_{\mathsf{\Pi}}$, define the instance $\hat{\mathbb{x}}_k$ and the corresponding proof $\hat{\pi}_k$.
5. For $i \in \mathsf{indexes}(\mathcal{J}_{k-1})$, run $p_i \leftarrow \mathcal{E}_{\mathsf{Com}}(\mathsf{c}_i)$
6. Return the forgery $(\hat{\mathbb{x}}_k, \hat{\pi}_k)$ and set the auxiliary input $\mathsf{aux}_{\mathcal{E}}$ as the adversary $\mathcal{A}_{\mathsf{\Pi}}$ does and include the polynomials $(p_i)_i$ extracted at the previous step.

By inspection, if the forgery of $\mathcal{A}_{\mathsf{\Pi}}$ passes the verification equation, then the forgery of $\mathcal{B}_{\mathsf{lin},k}$ passes the verification equation too.

Since the adversary $\mathcal{A}_{\mathsf{\Pi}}$ has not fully copied the transcript from any simulated proof, the random coin ρ_r computed by the verifier to verify the proof $\pi_{\mathsf{\Pi}}$ is, with overwhelming probability, a *fresh* coin, which implies that the simulator has never output a proof on it. Moreover, since the check $\hat{\mathbb{x}}_k$ is *focal*, the point $\hat{\mathbb{x}}_k.x$ is equal to $\tilde{v}_k(\rho_r)$ and $\mathsf{deg}_X(\tilde{v}) \geq 1$. The forgery of the reduction satisfies the Hash Check of Φ_{lin} because the reduction adds $\tilde{v}_k(X)$ to the auxiliary information of the RO query including all the commitments of $\hat{\mathbb{x}}_k$ at Item 3a. Moreover, when the distinguishing event between the two consecutive hybrids happens, the Partial-Extraction Check of the policy is satisfied:

- For $k = 1$ since, by definition, the index polynomials $(p_i)_{i \in n(0)}$ are honestly generated and therefore equal to what $\mathcal{E}_{\mathsf{Com}}$ extracts.
- For $k > 1$, because of the changes introduced in the hybrids $\mathbf{H}_{1,1}, \ldots \mathbf{H}_{1,k-1}$, the list of polynomials extracted in Item 5 are correctly extracted and included in the auxiliary information.

Finally, because of the compilation-safeness property, the linear independence check between the *left* polynomials of $\hat{\mathbb{x}}_k$ allows us to conclude that the forgery of $\mathcal{B}_{\mathsf{lin},k}$ matches the policy, while the distinguishing event asserts that the extractor fails. This bounds the probability of the distinguishing event to be at most ϵ_{lin}. □

Consider $\mathbf{H}_3$ that additionally returns 1 if the adversary $\mathcal{A}_{\mathsf{\Pi}}$ copies a simulated transcript up to r-th round from a simulated proof $\pi'_{\mathsf{\Pi}}$.

Lemma 4.7.3. $\Pr[\boldsymbol{H}_3] \leq \Pr[\boldsymbol{H}_2] + \epsilon_{\mathsf{del}}$

Proof. In this case, we have that π_{Π} and π'_{Π} use the same commitments $\vec{\mathsf{c}}$, the same messages $\vec{\pi}$ in the first r rounds, and there must be at least one commitment or a message in the last round that is different. Because of the change introduced in the previous hybrid, the distinguishing event focuses on the case in which the commitments sent in the last round can be extracted using $\mathcal{E}_{\mathsf{Com}}$, and moreover they satisfy the *focal* checks $\mathbb{x}_k$, for $k \in \mathcal{K}_2$.

However, by the uniqueness of PIOP with delegation phase property (see Definition 4.6.4), we have that the probability of the distinguishing event, which implies two different PIOP_2-transcripts for the same instance that differ on the first prover message (and polynomials) is $\epsilon_{\mathsf{del}} \in \mathsf{negl}(\log |\mathbb{F}|)$. □

Consider $\mathbf{H}_4$ that (similarly to $\mathbf{H}_2$) additionally returns 1 if

$$\exists k \in \mathcal{K}_1 : \sum_i G_i(p_{\gamma_1}(x), \ldots p_{\gamma_m}(x), x) \cdot p_{\beta_i}(x) \neq y \quad \vee$$
$$\exists j : \mathsf{c}_j \neq [p_j(s)]_1$$

where $\hat{\mathbb{x}}_k = \Big((\mathsf{c}_{\gamma_j})_j, (\mathsf{c}_{\beta_i})_i, (G_i)_i, x, y)\Big)$.

Lemma 4.7.4. $\Pr[\boldsymbol{H}_4] \leq \Pr[\boldsymbol{H}_3] + |\mathcal{K}_1| \cdot \epsilon_{\mathsf{lin}}$

Proof. The proof of this lemma proceeds almost identically to Lemma 4.7.1. The main difference is that, by the definition of compilation-safeness, a focal check $\hat{\mathbb{x}}_k$ with $k \in \mathcal{K}_1$ queries the polynomials at point $x = \tilde{v}_k(\rho_1, \ldots, \rho_{r-1})$ where $\mathsf{deg}_{X_{r-1}}(\tilde{v}_k) \geq 1$, thus we need to use the change introduced in $\mathbf{H}_3$ to make sure that the challenge ρ_{r-1} is different than in all the simulation proofs, and thus the evaluation point x is *fresh*, namely that the simulator of $\mathsf{CP}_{\mathsf{lin}}$ has never simulated a proof with x as evaluation point.

For completeness, we give the full proof of lemma.

We start noticing that, by our simplifying assumption on the order of the verifier's queries, we have that the indexes in $\mathcal{K}_1$ are consecutive numbers, in particular $\mathcal{K}_1 \coloneqq \{|\mathcal{K}_2|+1, \ldots, |\mathcal{K}_2|+|\mathcal{K}_1|\}$.

We prove this lemma through a series of hybrids. Let $\mathbf{H}_{3,|\mathcal{K}_2|} \equiv \mathbf{H}_3$. For any $k > |\mathcal{K}_2|$, let $\mathbf{H}_{3,k}$ be the same as $\mathbf{H}_{3,k}$, but that additionally returns 1 if:

$$\sum_i G_i(p_{\gamma_1}(x), \ldots, p_{\gamma_m}(x), x) p_{\beta_i}(x) \neq y \quad \vee$$
$$\exists i : \mathsf{c}_{\beta_i} \neq [p_{\beta_i}(s)]_1 \vee \exists j : \mathsf{c}_{\gamma_j} \neq \Big[p_{\gamma_j}(s)\Big]_1$$

where $\hat{\mathbb{x}}_k = \Big((\mathsf{c}_{\gamma_j})_j, (\mathsf{c}_{\beta_i})_i, (G_i)_i, x, y\Big)$. Finally, we have that $\mathbf{H}_4 \equiv \mathbf{H}_{3,|\mathcal{K}_2|+|\mathcal{K}_1|}$.

Let $\mathcal{I} \coloneqq \mathcal{CI}(\hat{\mathbb{x}}_k)$ be the *compiler-safe index set* of $\mathbb{x}_k$. We reduce to the $(\boldsymbol{\Phi}^{\mathcal{I},\nu}_{\mathsf{lin}}, \mathcal{F}_{\mathcal{I},\nu})$-simulation extractability of $\mathsf{CP}_{\mathsf{lin}}$. We make use of a reduction similar to the one used in the proof of Lemma 4.7.1.

Reduction $\mathcal{B}_{\mathsf{lin},k}(\mathsf{srs}, \mathsf{pp}_{\Phi_{\mathsf{lin}}})$

1. Run $\mathcal{A}_{\Pi}(\mathsf{srs})$.

2. Upon query $((\mathbb{i}, \mathbb{x}), \mathsf{aux})$ to the simulation oracle, run the same strategy of $\mathcal{S}_1$ defined in Fig. 4.5, namely using $\mathsf{CP}_{\mathsf{lin}}.\mathcal{S}_1$.
3. Upon query to $\mathcal{S}_2$:
   (a) If it can be parsed as a partial transcript $\mathtt{trns} = (\mathsf{vk}_{\mathbb{i}}, \mathbb{x}, \bar{\pi}_1, \ldots, \bar{\pi}_{r-1})$, derive the list $(\tilde{v}_{k'})_{k'} \leftarrow \tilde{\mathcal{V}}_2(\mathbb{F}, \mathbb{i}, \mathbb{x})$, forward the query to $\mathsf{CP}_{\mathsf{lin}}.\mathcal{S}_2$ adding the single-variable polynomial $(\tilde{v}_k(\rho_1, \ldots, \rho_{r-2}, X))_j$ to the auxiliary information.
   (b) Otherwise, simply forward the query to the simulator
4. Given the forgery $((\mathbb{i}, \mathbb{x}_{\Pi}), \pi_{\Pi})$ output by $\mathcal{A}_{\Pi}$, define the instance $\hat{\mathbb{x}}_k$ and the corresponding proof $\hat{\pi}_k$.
5. For $i \in \mathsf{indexes}(\mathcal{J}_{k-1})$, run $p_i \leftarrow \mathcal{E}_{\mathsf{Com}}(\mathsf{c}_i)$
6. Return the forgery $(\hat{\mathbb{x}}_k, \hat{\pi}_k)$ and set the auxiliary input $\mathsf{aux}_{\mathcal{E}}$ as the adversary $\mathcal{A}_{\Pi}$ does and include the polynomials $(p_i)_i$ extracted at the previous step.

Since the adversary $\mathcal{A}_{\Pi}$ has not copied, up to the r-th round, the transcript from any simulated proof, the last random coin ρ_{r-1} computed by the verifier to verify the proof π_{Π} is, with overwhelming probability, a *fresh* coin. Moreover, since the check $\hat{\mathbb{x}}_k$ is *focal*, the point $\hat{\mathbb{x}}_k.x$ is equal to $\tilde{v}_k(\rho_1, \ldots \rho_{r-1})$ and $\mathtt{deg}_{X_{r-1}}(\tilde{v}_k) \geq 1$. The forgery of the reduction satisfies the Hash Check of Φ_{lin} because the polynomial $\tilde{v}_k(\rho_1, \ldots, \rho_{r-2}, X)$ is added to the auxiliary information of the RO query including all the commitments of $\hat{\mathbb{x}}_k$. Similarly to the proof of Lemma 4.7.1, the Partial-Extraction Check of the policy is also satisfied.

Finally, because of the compilation-safeness property, the linear independence check between the *left* polynomials of $\hat{\mathbb{x}}_k$ allows us to conclude that the extractor would also extract its *core* and the *right* polynomials. By inspection, the list of polynomials extracted by $\mathcal{E}_{\mathsf{CP}_{\mathsf{lin}}}$ is equal to $(p_i)_i$ at the step 5 of the reduction. □ □

Consider $\mathbf{H}_5$ that additionally returns 1 if

$$\exists k \in [n_e] : \sum_i G_i(p_{\gamma_1}(x), \ldots p_{\gamma_m}(x), x) \cdot p_{\beta_i}(x) \neq y \quad \vee$$
$$\exists j : \mathsf{c}_j \neq [p_j(s)]_1$$

where $\hat{\mathbb{x}}_k = \Big((\mathsf{c}_{\gamma_j})_j, (\mathsf{c}_{\beta_i})_i, (G_i)_i, x, y\Big)$. Namely, if the polynomials extracted above do not satisfy the non-focal checks of $\mathcal{V}$.

Lemma 4.7.5. $\Pr[\boldsymbol{H}_5] \leq \Pr[\boldsymbol{H}_4] + (n_e - |\mathcal{K}_f|) \cdot \epsilon_{\mathsf{lin}}$

Proof. We can show a reduction to the $\Phi_{\mathsf{lin}+}$-simulation extractability of $\mathsf{CP}_{\mathsf{lin}}$. Similarly to the proof of Lemma 4.7.1, the reduction isolates the pair $(\hat{\mathbb{x}}_k, \pi_k)$ such that the distinguish event happens. Using the polynomials $p_i \leftarrow \mathcal{E}_{\mathsf{Com}}(\mathsf{c}_i)$ that are a valid witness for the *focal* checks because of the change introduced before, the reduction can create an algebraic inconsistent proof: given the result of Theorem 4.5.3, we can be bound the probability of such an event to ϵ_{lin}. □

Finally, we prove that the probability that $\mathcal{A}_{\mathsf{PIOP}}$ wins in $\mathbf{H}_5$ is equal to the probability that $\mathcal{A}_{\Pi}$ wins the strong simulation-extractibility experiment.

Lemma 4.7.6. $\Pr[\boldsymbol{H}_5] = \mathbf{Adv}^{\Phi_{\mathsf{SE}}\text{-}se}_{\mathcal{A}_\Pi,\mathcal{S},\mathcal{E}_\Pi}(\lambda)$

Proof. We now show that the probability that $\mathbf{H}_5$ outputs 1 is equal to the probability that the adversary $\mathcal{A}_\Pi$ wins the Φ_{SE}-se experiment against $\mathcal{E}_\Pi$. First, we notice that srs is generated by $\mathcal{S}(0, \mathsf{pp}_{\mathbb{G}})$ in both experiments. Because of the checks introduced in $\mathbf{H}_2$, $\mathbf{H}_4$ and $\mathbf{H}_5$, upon a valid forgery $((\mathbb{i}, \mathbb{x}_\Pi), \pi_\Pi)$, we have that:

- The proof π_Π cannot contain a simulated element (see Item 3 of $\mathcal{A}_{\mathsf{PIOP}}$). In fact, because of $\mathbf{H}_4$, all the polynomials can be extracted, and therefore cannot be simulated. Thus, all the RO queries of $\mathcal{A}_\Pi$ that constitute π_Π (namely the queries $q_1, \ldots, q_r$) are forwarded to the challenger in Item 3b of $\mathcal{A}_{\mathsf{PIOP}}$ (in particular, any of the queries is already answered by the programming of the RO by the simulator $\mathcal{S}_1$). This implies that the complete transcript sent by $\mathcal{A}_{\mathsf{PIOP}}$ is in the list $\mathsf{SeenStates}$.
- The extractor $\mathcal{E}_\Pi$ does not abort neither at Item 5 nor at Item 6 because the polynomials $(p_i)_i$ extracted by $\mathcal{E}_{\mathsf{Com}}$ satisfy the linearized checks $\hat{\mathbb{x}}_k$, for all $k \in [n_e]$. This implies that the extractor $\mathcal{E}_{\mathsf{PIOP}}$ in the $\mathbf{Exp}^{sr}_{\mathcal{A}_{\mathsf{PIOP}},\mathsf{PIOP}}$ in $\mathbf{H}_5$ is fed with the same polynomials extracted by $\mathcal{E}_\Pi$.

Thus, the decision bit in the state-restoration game is 1. □

Having bound the probability that the adversary $\mathcal{A}_\Pi$ wins the Φ_{SE}-se experiment against $\mathcal{E}_\Pi$ along the hybrids concludes the proof. □

Chapter 5

Non-malleable Virtual Machines

This chapter is extracted from "SNARKs for Virtual Machines are Non-Malleable", published in EUROCRYPT 2025.

5.1 Introduction

zkSNARKs for Virtual Machines (zkVMs) are behind the design of deployed systems with non-malleability requirements.

The approach we take in this chapter is:

(i) to analyze the simulation extractability of a *concrete, representative zkVM design* to use as a case study.

(ii) to provide, at the same time, *a set of methodological tools* for the study of the simulation extractability of zkVMs *in general*—that is, beyond our specific choice of zkVM construction in item *(i)*. In fact, as we elaborate on below, we will provide a set of technical results useful for *an even broader* family of SNARK constructions, namely *Lego-ish* SNARKs (which we define below).

Simulation extractability of Jolt. We will choose as a case study a design *loosely based on* Jolt, a lookup-singularity SNARK VM for the RISC-V instruction set, at the heart of which is Lasso, an argument for lookups with attractive efficiency features. This makes Jolt/Lasso a likely adoption in different settings in the near future [Tha22, Tha24c]. However, besides their efficiency, Jolt/Lasso constitute a natural choice since they together provide the first example of lookup-singularity SNARK VM having been concretely described and implemented. Finally, and crucially, Jolt [AST24] and Lasso [STW24], might be the most formal treatment of SNARKs for VMs in the literature at the present moment. This is important for us since otherwise we would not be able to carry out the type of formal analysis required by simulation extractability. To be more precise, the concrete design we will consider will not be exactly identical to that sketched in [AST24]. First off, the original description of Jolt and Lasso is not zero-knowledge. Since the framework of simulation extractability presupposes zero-knowledge, we have to naturally start from a zero-knowledge version of Lasso/Jolt. Second, for sake of

generality and simplicity, we will abstract out some parts of the Jolt design. At the high-level, Jolt runs a VM dividing it into three parts[1] each proven by a different "sub-SNARKs": instruction execution (via Lasso), instruction-fetching and memory-checking (both proven via Spartan-like proof systems [Set20]). In our concrete result (Corollary 5.8.1) we assume that instruction execution applies (our variant of) Lasso, while we abstract out the remaining sub-SNARK specifying what properties they need to satisfy in order for the final zkVM to be simulation-extractable.

zkVMs through the lens of modularity. Our discussion above hints to how it may be possible to approach the simulation extractability of zkVMs in general: since SNARKs for VMs lend themselves to *modular* designs, this is potentially something we can leverage[2]. Thus, on our way towards our goal in item *(ii)* above, we tackle a *more broadly interesting problem*: the non-malleability of *modular* (or *Lego-ish*) SNARKs [CFQ19], i.e. SNARKs that are obtained from the composition of several "sub-SNARKs", each possibly of a different design. In particular, we address this question:

What can we say about the non-malleability of a modular SNARK knowing that (some of) its building blocks are non-malleable?

Modular SNARKs have been identified as worth of a systematic investigation of their own because of their simplicity and efficiency [ABC+22, BCF+21, CFF+21, CFQ19]; general treatment of open problems in SNARKs designs—efficient distributed proving—have recently benefited from an explicit modular approach [RMH+24]. While we do have a general theoretical framework to reason about knowledge-soundness and zero-knowledge of Lego-ish SNARKs [CFQ19], to the best of our knowledge, no work prior to ours systematically studied the simulation extractability of modular SNARKs.

Challenges of Lego-ish SIM-EXT. We remark that composing non-malleable objects while maintaining their non-malleability does not come for free. For instance, as demonstrated in [FFK+23], there are *copy-paste* attacks when composing different Interactive Oracle Proofs (IOPs) (see Ben-Sasson, Chiesa and Spooner [BCS16]) into one simulation-extractable zk-SNARK. These attacks consider compositions of schemes for arbitrary relations without any shared knowledge. Briefly, our framework shows how to circumvent these attacks by "gluing" together the witnesses, either by considering a shared witness or by considering witnesses that are somehow logically linked (we will elaborate more in the next section and make these intuitions precise in our compilers in Section 5.7). To prove a statement composed of different relations, we will have to identify specific constraints for both the relations themselves and the sub-SNARKs used to prove each individual relation.

[1] We stress that in the Jolt paper, this distinction is sketched, and the reader can think of this paragraph as our own (intentionally fuzzy) paraphrase. A formal treatment of different components of a VM is highly dependent on the VM at hand. We will attempt a general formal treatment in Section 5.8.2.

[2] This modularity is not a mere technical artifact of the work in Jolt [AST24]. It has been used explicitly in other works [LZZ+24], and it is a natural design approach: different sub-components of VMs will have distinct features where sub-SNARKs of different designs will shine. Arguably, a modular design is already *explicitly* at the core of "lookup-singularity" SNARKs since their defining principle is to use a specialized SNARK (a lookup argument) for a specific component (instruction execution).

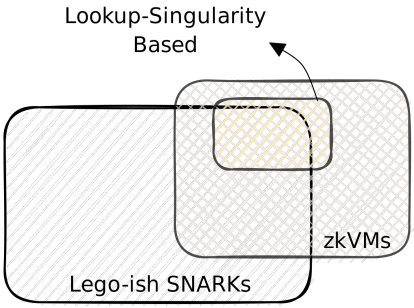


Figure 5.1: Intuition for why SIM-EXT results for Lego-ish SNARKs are useful for zkVMs in general: zkVMs in general lend themselves easily to a modular design; this is especially true for lookup-singularity based ones. See also Section 5.8.

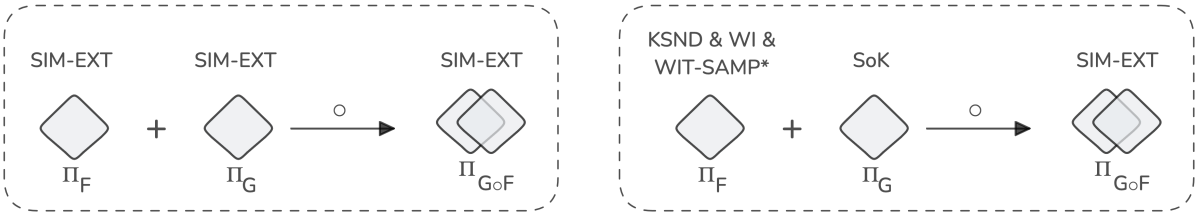


Figure 5.2: Informal description of two of our composition results for SIM-EXT of Lego-ish SNARKs (Theorem 5.7.1).

5.1.1 Our Contributions

5.1.1.1 Simulation Extractability of Joltish

Our main result consists in proving that, under the discrete log assumption, a lookup-based singularity zkVM based on Jolt—that we call Joltish—is simulation-extractable.

Our Joltish is based on our simulation-extractable lookup argument zkLasso, which makes Joltish a lookup-singularity zkVM. In the technical overview in Section 5.2 we give more details on how we obtain Joltish.

5.1.1.2 A toolbox for composition of simulation-extractable CP-SNARKs

A commit-and-prove argument of knowledge is an argument of knowledge where the witness is committed using a (non-interactive) commitment scheme. The work LegoSNARK of Campanelli, Fiore, and Querol [CFQ19] shows that commit-and-prove SNARKs are very useful for composing different SNARKs together in meaningful ways.

We show two compositions derived from commit-and-prove schemes that are simulation extractable. In particular, we provide two natural ways to compose schemes.

The first composition we consider is the conjunction of two relations. At first glance, given an argument for a relation $\mathcal{R}_F$ and an argument for a relation $\mathcal{R}_G$, we can realize an argument for "*their conjunction*" by running the two arguments independently. This composition is knowledge-sound; however, it is not simulation-extractable, as we can mount a copy-paste attack where the attacker knows a witness for the instance in $\mathcal{R}_F$ and uses a simulated proof for $\mathcal{R}_G$ (see [FFK+23] for more details). We avoid this attack by considering a conjunction of

relations where the committed witness is shared between the two instances. Given this, we show that if the two arguments (for $\mathcal{R}_F$ and $\mathcal{R}_G$) are simulation-extractable, then their composition is also simulation-extractable.

The second composition is what we call function composition. Consider a SNARK for correct function execution, namely, a SNARK that proves $F(\mathbb{x}, \mathbb{w}) = \mathbb{y}$ for a function F, with public input $\mathbb{x}$, private input $\mathbb{w}$, and output $\mathbb{y}$. Consider two commit-and-prove schemes: one that proves $F(\mathbb{x}_f, \mathbb{w}_f) = \mathbb{y}_f$, and a second that proves $G(\mathbb{x}_g, \mathbb{w}_g) = \mathbb{y}_g$ and let us call them Π_F and Π_G respectively. Now we can compose them together to prove $G \circ F(\mathbb{x}_f, \mathbb{x}_g, \mathbb{w}_f) := G(\mathbb{x}_g, F(\mathbb{x}_f, \mathbb{w}_f))$. The idea is to generate the first and second instances so that they share the commitment to $\mathbb{y}_f$, thus linking the private output of F with the private input of G. Also in this case, we can show that if the two commit-and-prove schemes are simulation-extractable, then their composition is also simulation-extractable.

These two results are rather straightforward and rely only on the fact that when the two schemes share knowledge, one cannot mount the trivial copy-paste attack described above.

We can actually improve the conditions of the results by assuming an extra property from one of the relations, which we refer formally as *efficient witness computability* (WIT-SAMP). Loosely speaking, this property states that we can find easily valid witnesses for the (non-committed part of the) instances. For example, in the functional composition, if the prover has the freedom to sample the commitment to $\mathbb{y}_f = \mathbb{w}_g$, then the zero-knowledge simulator for the composed scheme could sample a dummy input $(\mathbb{x}_f, \vec{0})$ for F, run the honest prover for Π_F, and simulate the proof for Π_G. Since the simulator for the composed scheme does not use the simulator for Π_F, we can (1) reduce the simulation extractability of the composed scheme to the knowledge soundness of Π_F, and (2) reduce the zero-knowledge of the composed scheme to the witness indistinguishability of Π_F. There is a caveat in this composition: Π_F could be re-randomizable, allowing the adversary to create a forgery for an instance where it has already seen a simulated proof (i.e., we can only prove *weak* simulation extractability for the composed scheme). However, we can address this issue, and prove *full* simulation extractability for the composed scheme, by assuming that Π_G is a signature-of-knowledge (SoK) and by *signing* the proof for Π_F using Π_G. In Fig. 5.2 we give a graphical representation of our results on generic composition of commit-and-prove SNARKs and summarize in the following informal version of Theorem 5.7.1.

Informal statement of Theorem 5.7.1. *There exists a black-box transformation from two SIM-EXT commit-and-prove SNARK Π_F, Π_G to a SIM-EXT conjunction (resp. composition) proof system $\Pi_{F\wedge G}$ (resp. $\Pi_{G\circ F}$). Moreover, there exists a black-box transformation to a SIM-EXT conjunction proof system $\Pi_{F\wedge G}$ (resp. for functions composition $\Pi_{G\circ F}$) from two commit-and-prove SNARKs Π_F, Π_G where (1) Π_F is KSND and (statistically) WI and $\mathcal{R}_F$ satisfies WIT-SAMP and (2) Π_G is a signature of knowledge.*

Recipes for parallelizable SIM-EXT SNARKs. A problem when using signature of knowledge is that we can call Π_G only after having computed the proof for Π_F, which forces sequentiality in the proof generation. To mitigate such a bottleneck, in Section 5.8.3 we describe a notion of signature of knowledge where, roughly speaking, the message can be fed at the very end of the prover's computations. We refer to this as a *signature of knowledge with delayed message*. We give two instantiations of SoK with delayed message. We show (1) that the classical Fiat-Shamir approach for signature of knowledge can be adapted to the delayed message setting extending the results on Fiat-Shamir-based simulation-extractable argument [DG23]

and (2) we give a black-box construction of signature-of-knowledge with delayed message from (classical) signature-of-knowledge and one-time signatures.

5.1.1.3 Other contributions

At the technical basis of our results on the non-malleable zkVMs lies a series of contributions that we are going to present in more detail in the next section. First, we give a zero-knowledge version of Lasso and provide the analysis of its simulation extractability. Second, we revisit the technical results of [DG23], weakening their requirements and achieving tighter bounds for Spartan and Bulletproofs. Finally, we give a proof of the simulation extractability of `HyraxPC`, which may be of independent interest.

5.1.2 Organization of the chapter

In Section 5.2 we elaborate more on our results and the technical challenges that we had to overcome. In Section 5.3, we give a formal definition of interactive and non-interactive arguments in the ROM. In Section 5.4 we show an efficient *tree-builder* and we prove the knowledge soundness of the arguments compiled using the Fiat-Shamir transform. In Section 5.5 we introduce the Hyrax polynomial commitment and we prove that it is simulation-extractable. In Section 5.6, we introduce the Lasso lookup argument and we show how to make it zero-knowledge and simulation-extractable. Finally, in Section 5.7 we give our toolbox on composition of simulation-extractable CP-SNARKs and we elaborate on the applications to zkVMs in Section 5.8.

5.2 A Technical Overview of Our Results

Simulation extractability of zkLasso. The technical core of our contribution is providing a simulation-extractable indexed lookup argument derived from Lasso. We take the work of [DG23] as our starting point. They prove the simulation extractability of (zero-knowledge variants of) schemes such as Bulletproofs and Spartan. Their work follows the results of simulation extractability for Fiat-Shamir based arguments inspired by the work of Faust *et al.* [FKMV12] and further investigated in [GKK+22, GOP+22, KPT23].

Their approach works in three steps which together provide simulation extractability: *(i)* have ZK version of the protocols;[3] *(ii)* prove that all the inner (sub)protocols are *computational*[4] special-sound, i.e., it is possible to extract a witness from a sufficient number of valid proofs and whose transcript possibly satisfies some additional *predicate*; *(iii)* proving that for a specific k (where k is a round index) the protocol satisfies two properties referred as k-ZK and k-unique response (k-UR for short). k-ZK restricts the ZK simulator by allowing it to reprogram the random oracle only at the k-th round. k-UR states that the malicious prover's responses are uniquely determined after the k-th round.

To achieve step *(i)*, Dao and Grubbs need to replace all the occurrences of the inner protocols, such as the sum-check-based reductions, with their *blinded* versions. For example, if

[3] The usual notion of simulation extractability makes sense for ZK protocols only.

[4] If the extractor fails to extract a witness, then we argue that the malicious prover is able to break some computationally-hard problem, e.g., finding a nontrivial discrete log relation between the Pedersen generators.

we consider the classical sum-check protocol which eventually evaluates on a random point $\vec{x}$ defined by the verifier's challenges a committed polynomial f, the blinded counterpart would instead commit to $f(\vec{x})$, for example using Pedersen, and then show in zero-knowledge that such a commitment opens to the evaluation of f on $\vec{x}$. Thus, the scheme *blinds* the value y which might leak information about the witness.

While several of the building blocks of Lasso are common to those of Spartan, and we naturally use some of the same "low-level" technical tools, our analysis diverges substantially from that of [DG23] and requires developing some more machinery. More in detail, we follow [DG23] and substitute the sub-protocols with their blinded versions. To do so, we need to define a blinded version of the grand product argument due to [Tha13] and prove it computational special-sound. Once done that, we need a stronger analysis of the computational special soundness of the hash-based multi-set fingerprinting used in Spartan. Specifically, Lasso and Spartan use Spark as their underlying (sparse) polynomial commitment scheme; however, while in Spartan some of the sparse polynomials are committed honestly by the verifier, in Lasso these polynomials are committed by the untrusted prover. Crucially, in our case these sparse polynomials encode the matrix of the lookup indexes, i.e., the witness we wish to extract from the proof of the adversary, and this discrepancy introduces non-trivial differences between our work and [DG23] when analyzing the computational special soundness.

A second component of both Lasso and Spartan is (yet another) polynomial commitment called `HyraxPC` [WTs+17]. We improve the analysis for `HyraxPC`. Specifically, digging into the technical details of [DG23], to prove computational special soundness of Hyrax, we need first to define a tree of transcripts where the edges of each node satisfy a set of constraints that Dao and Grubbs formalize through a set of *predicates*. We show that we need to introduce one more predicate to fix the proof of computational special soundness of `HyraxPC`. Moreover, we additionally prove that `HyraxPC` achieves k-ZK and k-UR, and thus, as additional result, we can prove that this polynomial commitment is simulation-extractable.

By revisiting the techniques of [DG23], we also introduce some improvements that directly apply to Spartan and Bulletproofs, as well as to Lasso. First, we design a (slightly) tighter blinded sum-check protocol that only relies on the simple *distinctness* predicate, and for which it is sufficient to use the tree-builder of Attema *et al.* [ACK21]. Second, and more importantly, we achieve a tighter bound in our extractor (cf. Theorem 5.4.1) avoiding a loss quadratic in the number of the prover's queries and by solving a problem left open in the previous work.

We observe that our approach is still rewinding-based and so the provable SIM-EXT security we get is "low" in terms of security bits, however this seems inherent to this type of analysis.

From zkLasso to Joltish. We provide a model for arguments of knowledge for virtual machine execution. While similar formalizations exist in the literature [BCG+13, BCG+18, DOTV22, ZGK+18], our framework focuses on abstracting zkVMs based on the lookup singularity. We isolate the logical components in the VM that lookup argument can handle from the rest and demonstrate that our compiler for conjunction, described in Section 5.7, is sufficient to achieve simulation-extractable zkVMs. More in detail, we adopt an indirect way to achieve such a formalization: we define a commit-and-prove relation $\mathcal{R}^\star$ as the series of logical and memory constraints and checks to perform to the trace of the *program execution* which, together with the correct *instructions execution* handled by the lookup argument, prove correct program execution. This abstraction results in a conjunction of a scheme for $\mathcal{R}^\star$ and a lookup argument. Thus, we can use our general non-malleable composition results (see the informal theorem at

page 122 and the associated Theorem 5.7.1). In particular, we can leverage on a simulation-extractable lookup argument to weaken the necessary security properties of the scheme for $\mathcal{R}^\star$. To do so, we show that $\mathcal{R}^\star$ is WIT-SAMP, by showing how to derive a valid trace of a program execution that uses an invalid instruction set. As a consequence, the scheme for $\mathcal{R}^\star$ needs only to be WI and knowledge sound, which open the doors to many instantiations.

Next, we demonstrate how to integrate our zkLasso into the framework, resulting in a broad class of zkVMs that, as we argue, includes Joltish, our zero-knowledge variant of Jolt [AST24]. This task is easier since we can use the knowledge-soundness results for the scheme(s) for $\mathcal{R}^\star$ of [AST24]. We emphasize that our composition theorem for zkVMs, Theorem 5.8.1, is general and allows for the replacement of components in Joltish with different SNARKs, which is why we refer to a large class of zkVMs.

5.3 Arguments in the ROM

In this chapter, we focus on SNARKs whose setup is *transparent*. In this setting, the simulator is *trapdoor-less*, i.e., it can only reprogram the random oracle to simulate the proofs, and may *rewind* the adversary. For this reason, we prefer to slightly depart from the syntax and definition of NIZKs given in Section 2.4.3.1.

5.3.0.1 Interactive Arguments

A public-coin interactive argument for a relation $\mathcal{R}$ is a tuple of PPT algorithms $\Pi \coloneqq (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ where:

$\mathsf{Setup}(1^\lambda, \mathsf{pp}_\mathcal{G}) \to \mathsf{pp}$**:** outputs parameters pp given global parameters $\mathsf{pp}_\mathcal{G}$

$\langle \mathcal{P}(\mathbb{w}), \mathcal{V} \rangle(\mathsf{pp}, \mathbb{x}) \to \{0,1\}$**:** a public-coin interactive protocol whereby the prover $\mathcal{P}$, holding a witness $\mathbb{w}$, interacts with the verifier $\mathcal{V}$ on common input $(\mathsf{pp}, \mathbb{x})$ to convince $\mathcal{V}$ that $(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$. At the i-th round, $\mathcal{V}$ samples its message uniformly at random from the challenge space $\mathcal{C}_i$. At the end, $\mathcal{V}$ outputs a bit to accept or reject.

We consider interactive arguments that satisfy the standard properties completeness, knowledge soundness and honest-verifier zero-knowledge.

Completeness: For any adversary $\mathcal{A}$ we have that:

$$\Pr\left[\begin{array}{l|l} (\mathsf{pp}, \mathbb{x}, \mathbb{w}) \notin \mathcal{R} \vee & \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{pp}_\mathbb{G}) \\ \langle \mathcal{P}(\mathbb{w}), V \rangle(\mathsf{pp}, \mathbb{x}) = 1 & (\mathbb{x}, \mathbb{w}) \leftarrow \mathcal{A}(\mathsf{pp}) \end{array}\right] = 1$$

Knowledge-Soundness: There exists an EPT extractor $\mathcal{E}$ such that for any stateful PPT adversary $\mathcal{P}^*$:

$$\Pr\left[b = 1 \wedge (\mathsf{pp}, \mathbb{x}, \mathbb{w}) \notin \mathcal{R} \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{pp}_\mathbb{G}) \\ (\mathbb{x}, \mathsf{st}_{\mathcal{P}^*}) \leftarrow \mathcal{P}^*(\mathsf{pp}) \\ b \leftarrow \langle \mathcal{P}^*(\mathsf{st}_{\mathcal{P}^*}), \mathcal{V} \rangle(\mathsf{pp}, \mathbb{x}) \\ \mathbb{w} \leftarrow \mathcal{E}^{\mathcal{P}^*}(\mathsf{pp}, \mathbb{x}) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

where $\mathcal{E}$ gets black-box access to each of the next-message functions of $\mathcal{P}^*$ in the interactive protocol and can rewind $\mathcal{P}^*$ to any point in the interaction.

Game $\mathsf{KS}^{\mathcal{P}^*}_{0,\Pi_{\mathsf{FS}},\mathcal{R}}(\lambda)$	Game $\mathsf{KS}^{\mathcal{E},\mathcal{P}^*}_{1,\Pi_{\mathsf{FS}},\mathcal{R}}(\lambda)$
$\mathsf{pp} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda, \mathsf{pp}_{\mathbb{G}})$	$\mathsf{pp} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda, \mathsf{pp}_{\mathbb{G}})$
$(\mathbb{x}, \pi) \leftarrow \mathcal{P}^{*\mathsf{H}}(\mathsf{pp})$	$(\mathbb{x}, \pi) \leftarrow \mathcal{P}^{*\mathsf{H}}(\mathsf{pp})$
$b \leftarrow \mathcal{V}^{\mathsf{H}}(\mathsf{pp}, \mathbb{x}, \pi)$	$b \leftarrow \mathcal{V}^{\mathsf{H}}(\mathsf{pp}, \mathbb{x}, \pi)$
return b	$\mathbb{w} \leftarrow \mathcal{E}^{\mathcal{P}^*}(\mathsf{pp}, \mathbb{x}, \pi)$
	return $b \wedge (\mathsf{pp}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$

Figure 5.3: Knowledge soundness security games. The extractor $\mathcal{E}$ is given black-box access to $\mathcal{P}^*$, it simulates H and and can rewind $\mathcal{P}^*$ to any point.

Honest-Verifier Zero-Knowledge: There exists a PPT simulator $\mathcal{S}$ such that for all $\mathsf{pp} \leftarrow \mathsf{Setup}(\mathsf{pp}_{\mathbb{G}})$ and $(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$, the following distributions are statistically indistinguishable:

$$\{\mathrm{View}_{\mathcal{V}}\langle \mathcal{P}(\mathbb{w}), \mathcal{V}\rangle(\mathsf{pp}, \mathbb{x})\} \approx_s \{\mathcal{S}(\mathsf{pp}, \mathbb{x})\}$$

where $\mathrm{View}_{\mathcal{V}}\langle \mathcal{P}(\mathbb{w}), \mathcal{V}\rangle(\mathsf{pp}, \mathbb{x})$ denotes the view of the verifier, consisting of the transcript and its own randomness.

5.3.1 Non-Interactive Arguments

A non-interactive argument (in the ROM) for a relation $\mathcal{R}$ is a tuple of PPT algorithms $\Pi := (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ where: $\mathsf{Setup}(\mathsf{pp}_{\mathbb{G}}) \to \mathsf{pp}$ generates the public parameters $\mathcal{P}^{\mathsf{H}}(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \to \pi$ generates a proof π $\mathcal{V}^{\mathsf{H}}(\mathsf{pp}, \mathbb{x}, \pi) \to b$ checks if a proof is valid or not and outputs a bit $b \in \{0, 1\}$ and H is a random oracle.[5]

We consider non-interactive arguments that satisfy the following two properties.

Completeness: For any adversary $\mathcal{A}$ we have that:

$$\Pr\left[\begin{array}{l|l} (\mathsf{pp}, \mathbb{x}, \mathbb{w}) \notin \mathcal{R} \vee & \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{pp}_{\mathbb{G}}) \\ \mathcal{V}^{\mathsf{H}}(\mathsf{pp}, \mathbb{x}, \pi) = 1 & (\mathbb{x}, \mathbb{w}) \leftarrow \mathcal{A}^{\mathsf{H}}(\mathsf{pp}) \\ & \pi \to \mathcal{P}^{\mathsf{H}}(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \end{array}\right] = 1$$

Knowledge-Soundness: There exists an EPT extractor $\mathcal{E}$ such that for any stateful PPT adversary $\mathcal{P}^*$, the following probability is negligible in λ:

$$\mathbf{Adv}^{\mathsf{KS}}_{\Pi_{\mathsf{FS}},\mathcal{R}}(\mathcal{E}, \mathcal{P}^*) := \left|\Pr\left[\mathsf{KS}^{\mathcal{P}^*}_{0,\Pi_{\mathsf{FS}},\mathcal{R}}(\lambda)\right] - \Pr\left[\mathsf{KS}^{\mathcal{E},\mathcal{P}^*}_{1,\Pi_{\mathsf{FS}},\mathcal{R}}(\lambda)\right]\right|$$

and the knowledge soundness games are defined in Fig. 5.3.

Zero-Knowledge: There exists a PPT simulator $\mathcal{S}$ such that for $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{Setup}(\mathsf{pp}_{\mathbb{G}})$ and any unbounded adversary $\mathcal{A}$:[6]

$$\Pr\left[\mathcal{A}^{\mathsf{H}(\cdot),\mathcal{P}(\mathsf{pp},\cdot,\cdot)}(1^\lambda) = 1\right] \approx_s \Pr\left[\mathcal{A}^{\mathsf{H}(\cdot),\mathcal{S}^{\mathsf{RePro}}(\mathsf{pp},\cdot,\cdot)}(1^\lambda) = 1\right]$$

[5] For public-coin $(2r+1)$-message interactive arguments with challenge spaces $\mathcal{C}_1, \ldots, \mathcal{C}_r$, we actually need r independent random oracles $\mathsf{H}_i \colon \{0,1\}^* \to \mathcal{C}_i$ with $i \in [r]$. For simplicity, we denote these by a single random oracle H, and it will be clear from context which random oracle is being used in a given round.

[6] Zero-knowledge is a security property that is only guaranteed for valid statements in the language, hence $\mathcal{A}$ never queries $\mathcal{P}/\mathcal{S}$ with a pair $(\mathbb{x}, \mathbb{w})$ such that $(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \notin \mathcal{R}$.

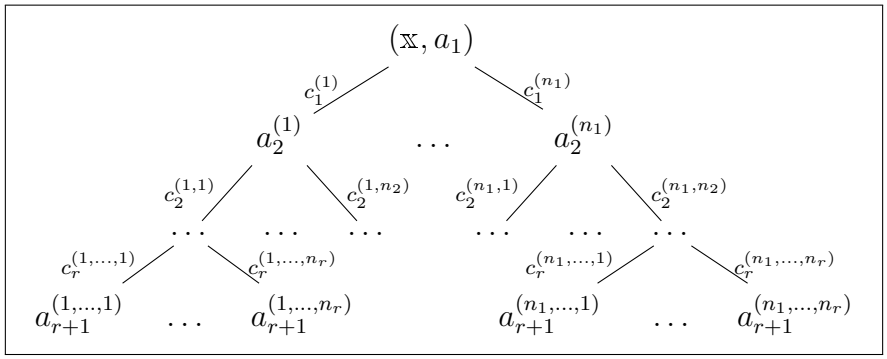


Figure 5.4: An $(n_1, \ldots, n_r)$-tree of transcripts for a $(2r+1)$-message public-coin protocol.

where RePro is an oracle that on input a pair (a, b) reprograms $\mathsf{H}(a) := b$.

We stress that zero-knowledge is defined in a model where the random oracle is *explicitly-programmable* [Wee09] by the simulator: in particular, $\mathcal{S}$ can reprogram the random oracle H (using RePro).

To turn public-coin interactive arguments into their non-interactive versions, we can employ the Fiat-Shamir (FS) transform in a setting where $\mathcal{P}$ and $\mathcal{V}$ have black-box access to a random oracle H. We use Π_{FS} to denote the non-interactive argument derived by applying the FS transform to the argument Π.

5.3.2 Tree of Transcripts

A $(n_1, \ldots, n_r)$-tree of transcripts for a $(2r+1)$-message public-coin protocol is a set of $\prod_{i \in [r]} n_i$ transcripts arranged in the following tree structure (see Fig. 5.4 for a graphical illustration):

- The nodes in this tree correspond to the prover's messages and the edges correspond to the verifier's challenges.
- Every node at depth i has precisely n_i children.
- Every transcript corresponds to exactly one path from the root to a leaf.

This notion, introduced by [ACK21], was later generalized by [DG23] to support custom predicates for the verifier challenges. In particular, in the generalization of [DG23], the edges (i.e., the verifier's challenges) of each node need to be distinct, and they also need to jointly satisfy a predicate ϕ_i where i is the depth of their corresponding node. In this work, we consider only the following predicates:

- $\phi_{\pm}$ that on input n field elements $(c_1, \ldots, c_n)$ returns 1 if and only if for all $i \in [n]$, there is not $j \neq i$ such that $c_i + c_j = 0$. We use the shortcut $n_{\pm}$ to indicate a node supporting this predicate.
- $\phi_{:k}$ that on input n challenges $(c_1, \ldots, c_n) \in \mathbb{F}^{n \cdot m}$ returns 1 if and only if all the inputs have different prefixes of length k. We use the shortcut $n_{:k}$ to indicate a node supporting this predicate.

We give a formal definition hereafter.

Game $\mathsf{SS}^{\mathcal{TE},\mathcal{A}}_{\Pi,\mathcal{R},(\vec{\phi},\vec{n})}(\lambda)$

$\mathsf{pp} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda, \mathsf{pp}_\mathbb{G})$

$(\mathbb{x}, \mathsf{T}) \leftarrow \mathcal{A}(\mathsf{pp})$

$\mathbb{w} \leftarrow \mathcal{TE}(\mathsf{pp}, \mathbb{x}, \mathsf{T})$

return $(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \notin \mathcal{R} \wedge \mathsf{IsAccepting}((\vec{\phi}, \vec{n}), \mathsf{pp}, \mathbb{x}, \mathsf{T})$

Figure 5.5: Computational Special Soundness security game.

Definition 5.3.1 (Tree of Transcripts)**.** *Let* Π *be a* $(2r+1)$*-message public-coin interactive argument for a relation* $\mathcal{R}$*, with challenge spaces* $\mathcal{C}_1, \ldots, \mathcal{C}_r$*. Let* $\vec{n} := (n_1, \ldots, n_r) \in \mathbb{N}^r$*, and let* $\vec{\phi} := (\phi_1, \ldots, \phi_r)$ *with* $\phi_i \colon \mathcal{C}_i \to \{0,1\}$*, for all* $i \in [r]$*, we say that* T *is a* $(\vec{\phi}, \vec{n})$*-tree of accepting transcripts for* pp *if:*

1. T *is a tree of depth* $r+1$*,*
2. *For all* $i \in [r+1]$*, each vertex at depth* i *is labeled with a prover's message* a_i*, and if* $i \leq r$ *it has exactly* n_i *outgoing edges to its children, with each edge labeled with a verifier's challenge* $c_{i,1}, \ldots, c_{i,n_i} \in \mathcal{C}_i^{n_i}$*, satisfying* $\phi_i(c_{i,1}, \ldots, c_{i,n_i}) = 1$*. Additionally, the root label is prepended with* $\mathbb{x}$ *(its label becomes* $(\mathbb{x}, a_1)$*),*
3. *The labels on any root-to-leaf path form a valid input-transcript pair* $(\mathbb{x}, \mathrm{tr})$*.*

We say that T *is* accepting *with respect to an input-transcript pair* $(\mathbb{x}, \mathrm{tr})$ *if* $(\mathbb{x}, \mathrm{tr})$ *corresponds to the left-most path of* T*. We define an acceptance predicate* $\mathsf{IsAccepting}((\vec{\phi}, \vec{n}), \mathsf{pp}, \mathbb{x}, (\pi,)\mathsf{T})$ *to check whether* T *is a* $(\vec{\phi}, \vec{n})$*-tree of accepting transcripts for* pp *and* $\mathbb{x}$*, and optionally* π*.*

We now define computational special soundness that essentially guarantees that there exists a tree-extractor algorithm $\mathcal{TE}$ that, given as input a tree of accepting transcripts produced by an efficient adversary, outputs a valid witness with high probability.

Definition 5.3.2 (Computational Special Soundness)**.** *Let* Π *be a* $(2r+1)$*-message public-coin interactive argument for a relation* $\mathcal{R}$ *with challenge spaces* $\mathcal{C}_1, \ldots, \mathcal{C}_r$*. For any* $\vec{n} := (n_1, \ldots, n_r) \in \mathbb{N}_r$ *and any* $\vec{\phi} := (\phi_1, \ldots, \phi_r)$ *with* $\phi_i \colon \mathcal{C}_i^{n_i} \to \{0,1\}$*, we say* Π *is* $(\vec{\phi}, \vec{n})$*-computational special sound if there exists a PPT tree-extraction algorithm* $\mathcal{TE}$ *such that for every EPT adversary* $\mathcal{A}$*, the following probability is negligible in* λ*:*

$$\mathbf{Adv}^{\mathsf{SS}}_{\Pi,\mathcal{R},(\vec{\phi},\vec{n})}(\mathcal{TE}, \mathcal{A}) := \Pr\left[\mathsf{SS}^{\mathcal{TE},\mathcal{A}}_{\Pi,\mathcal{R},(\vec{\phi},\vec{n})}(\lambda)\right]$$

and the special soundness game is defined in Fig. 5.5

Attema *et al.* prove the existence of an efficient *tree-builder* algorithm that can generate $\vec{n}$-trees of accepting transcripts having oracle access to a (malicious) prover $\mathcal{P}^*$. This result was later generalized by [DG23] to support partition predicates; in Section 5.4 we show how to adapt their result to achieve tighter bounds for the predicates needed to instantiate Spartan [Set20] and Bulletproofs [BBB+18].

Game $\mathsf{SE}^{\mathcal{S},\mathcal{P}^*}_{0,\Pi_{\mathsf{FS}},\mathcal{R}}(\lambda)$	Game $\mathsf{SE}^{\mathcal{E},\mathcal{S},\mathcal{P}^*}_{1,\Pi_{\mathsf{FS}},\mathcal{R}}(\lambda)$
$\mathsf{pp} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda, \mathsf{pp}_{\mathbb{G}})$	$\mathsf{pp} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda, \mathsf{pp}_{\mathbb{G}})$
$(\mathbb{x}, \pi) \leftarrow \mathcal{P}^{*\mathsf{H},\mathcal{S}}(\mathsf{pp})$	$(\mathbb{x}, \pi) \leftarrow \mathcal{P}^{*\mathsf{H},\mathcal{S}}(\mathsf{pp})$
$b \leftarrow \mathcal{V}^{\mathsf{H}'}_{\mathsf{FS}}(\mathsf{pp}, \mathbb{x}, \pi)$	$b \leftarrow \mathcal{V}^{\mathsf{H}'}_{\mathsf{FS}}(\mathsf{pp}, \mathbb{x}, \pi)$
return $b \wedge (\mathbb{x}, \pi) \notin \mathcal{Q}_{\mathcal{S}}$	$\mathbb{w} \leftarrow \mathcal{E}^{\mathcal{P}^*}(\mathsf{pp}, \mathbb{x}, \pi)$
	return $b \wedge (\mathbb{x}, \pi) \notin \mathcal{Q}_{\mathcal{S}} \wedge (\mathsf{pp}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$

Figure 5.6: Simulation extractability security games. $\mathcal{S}$ returns a proof π upon an input $\mathbb{x}$ (and may reprogram the random oracle), while $\mathcal{Q}_{\mathcal{S}}$ records all the pairs $(\mathbb{x}, \pi)$ queried by $\mathcal{P}^*$. H' denotes the modified RO after all the proof simulation queries. $\mathcal{E}$ is given black-box access to $\mathcal{P}^*$; in particular, it simulates H and $\mathcal{S}$ for $\mathcal{P}^*$ and can rewind $\mathcal{P}^*$ to any point in its execution (with same initial randomness).

5.3.3 Simulation extractability

Simulation extractability requires that extractability holds even when the malicious prover is given access to simulated proofs, possibly for *false* statements.

Definition 5.3.3 (Simulation extractability)**.** *Let $\Pi := (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ be a public-coin zero-knowledge interactive argument for relation $\mathcal{R}$ with associated NIZK $\Pi_{\mathsf{FS}} := (\mathsf{Setup}, \mathcal{P}_{\mathsf{FS}}, \mathcal{V}_{\mathsf{FS}})$. We say Π_{FS} is simulation extractable (with respect to a simulator $\mathcal{S}$) if there exists an EPT extractor $\mathcal{E}$ such that for every PPT adversary $\mathcal{P}^*$, the following probability is negligible in λ:*

$$\mathbf{Adv}^{\mathsf{SIM-EXT}}_{\Pi_{\mathsf{FS}},\mathcal{R}}(\mathcal{S}, \mathcal{E}, \mathcal{P}^*) := \left|\Pr\left[\mathsf{SE}^{\mathcal{S},\mathcal{P}^*}_{0,\Pi_{\mathsf{FS}}}(\lambda)\right] - \Pr\left[\mathsf{SE}^{\mathcal{E},\mathcal{S},\mathcal{P}^*}_{1,\Pi_{\mathsf{FS}}}(\lambda)\right]\right|$$

and the security games are defined in Fig. 5.6.

Hereafter, we introduce two more properties, namely k-zero-knowledge and k-unique response. Roughly speaking, the former notion captures zero-knowledge when the simulator is only allowed to reprogram the random oracle in the k-th round, while the latter states that the malicious prover's responses are uniquely determined after the k-th round. These two properties together with knowledge-soundness imply simulation extractability [GKK+22, DG23].

Definition 5.3.4 (k-Zero-Knowledge, [DG23])**.** *Let $\Pi := (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ be a $(2r+1)$-message public-coin interactive. We say that Π_{FS} satisfies (perfect) k-zero-knowledge, for some $k \in [r]$, if there exists a zero-knowledge simulator $\mathcal{S}_{\mathsf{FS},k}$ that only needs to program the random oracle in round k, and whose output is identically distributed to that of honestly generated proofs.*

Definition 5.3.5 (k-Unique Response, [DG23])**.** *Let $\Pi := (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ be a $(2r+1)$-message public-coin interactive argument. We say that Π_{FS} satisfies k-unique response, for some $k \in [r]$, if for every PPT adversary $\mathcal{A}$:*

$$\Pr\left[\, b \wedge b' \wedge \pi \neq \pi' \wedge \pi|_k = \pi'|_k \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{pp}_{\mathbb{G}}) \\ (\mathbb{x}, \pi, \pi', c) \leftarrow \mathcal{A}^{\mathsf{H}}(\mathsf{pp}) \\ b \leftarrow \mathcal{V}^{\mathsf{H}[(\mathsf{pp},\mathbb{x},\pi|_k)\to c]}_{\mathsf{FS}}(\mathsf{pp}, \mathbb{x}, \pi) \\ b' \leftarrow \mathcal{V}^{\mathsf{H}[(\mathsf{pp},\mathbb{x},\pi'|_k)\to c]}_{\mathsf{FS}}(\mathsf{pp}, \mathbb{x}, \pi') \end{array} \right] \in \mathsf{negl}(\lambda)$$

where $\mathsf{H}[x \to c]$ denotes the RO when the input x is reprogrammed to output c.

Theorem 5.3.1 ([DG23]). *Let Π be a $(2r+1)$-message public-coin interactive argument. If Π_{FS} is knowledge-sound and there is $k \in [r]$ such that Π_{FS} satisfies k-zero-knowledge and k-unique response, then Π_{FS} is simulation extractable.*

5.4 A Tree Builder for Efficiently-decidable Partitions

A technical tool we leverage is an efficient *tree-builder* to prove the knowledge soundness of computational special sound arguments compiled using the Fiat-Shamir transform, that was studied in the work of [DG23] in the wake of the results of [GKK+22, GOP+22].

We start by introducing the notions of an abstract adversary and an abstract tree of transcripts that can be defined independently of any interactive argument Π.

Definition 5.4.1 (Abstract Adversary). *Let $\mathcal{S}_1, \ldots, \mathcal{S}_r$ be finite sets and let $\mathsf{H} := (\mathsf{H}_1, \ldots, \mathsf{H}_r)$ be a collection of random oracles $\mathsf{H}_i \colon \{0,1\}^* \to \mathcal{S}_i$. An r-round and Q-query random oracle adversary $\mathcal{A}$ against $(\mathcal{S}_1, \ldots, \mathcal{S}_r)$ is a deterministic adversary having oracle access to H, making at most Q total accesses to these random oracles, and returning $((a_1, \ldots, a_{r+1}), v)$ where $(a_i)_{i\in[r]}$ are strings and $v \in \{0,1\}$. The success probability of $\mathcal{A}$ is $\Pr\left[v = 1 \mid ((a_1, \ldots, a_r), v) \leftarrow \mathcal{A}^{\mathsf{H}}\right]$ and this probability is defined over the randomness of choosing H.*

Definition 5.4.2 (Abstract Tree of Transcripts). *Let $\mathcal{S}_1, \ldots, \mathcal{S}_r$ be finite sets, $\mathcal{A}$ be any abstract adversary against $\mathcal{S}_1, \ldots, \mathcal{S}_r$, and $\vec{n} := (n_1, \ldots, n_r) \in \mathbb{N}^r$. An $\vec{n}$-abstract tree of transcripts T for $\mathcal{A}$ and $\mathsf{H} := (\mathsf{H}_1, \ldots, \mathsf{H}_r)$ is a labeled $\vec{n}$-tree where:*

- *Each vertex at depth $i \in [r+1]$ is labeled with a message a_i*
- *Each of the n_i edges coming from a vertex at depth $i \in [r]$ is labeled with a different element $s \in \mathcal{S}_i$*
- *For any root-to-leaf path, if the edges are labeled $(s1, \ldots, s_r)$ and the vertices are labeled $(a_1, \ldots, a_{r+1})$ then $((a_1, \ldots, a_{r+1}), 1) \leftarrow \mathcal{A}^{\mathsf{H}'}$ where $\mathsf{H}' := (\mathsf{H}_1[a_1 \to s_1], \ldots, \mathsf{H}_r[(a_1, \ldots, a_r) \to s_r])$.*

Let Π be a $(2r+1)$-message public-coin interactive argument with challenge sets $\mathcal{C}_1, \ldots, \mathcal{C}_r$. From any deterministic adversary $\mathcal{P}^*$ against the knowledge-soundness of Π_{FS}, we can build an abstract adversary $\mathcal{A}$ against the sets $\mathcal{C}_1, \ldots, \mathcal{C}_r$ by running $(x, (a_1, \ldots, a_{r+1})) \leftarrow \mathcal{P}^{*\mathsf{H}}(\mathsf{pp})$ (with pp hard-coded) and also $v \leftarrow \mathcal{V}^{\mathsf{H}}_{\mathsf{FS}}(\mathsf{pp}, \mathbb{x}, \pi)$. $\mathcal{A}$ then outputs $(((\mathbb{x}, a_1), a_2, \ldots, a_{r+1}), v)$. An $\vec{n}$-tree of accepting transcripts for $(\mathsf{pp}, \mathbb{x}, \pi)$ can be seen as an $\vec{n}$-abstract tree of transcripts for $\mathcal{A}$.

Definition 5.4.3 (Partition Predicates). *Let $\mathcal{C} := \bigcup_{i\in[C]} \mathcal{C}^{(i)}$ be a partition $\mathcal{P}$ of a set $\mathcal{C}$ into C blocks. We assume the partition is efficient, i.e. given an index $i \in [C]$, we can enumerate the set $\mathcal{C}^{(i)}$ in polynomial time. For $n \in \mathbb{N}$, we define the corresponding partition predicate $\phi_{\mathcal{P},n} \colon \mathcal{C}^n \to \{0,1\}$ to output 1 on input $(c_1, \ldots, c_n)$ if and only if $c_1, \ldots, c_n$ belong to distinct blocks of $\mathcal{C}$.*

We consider the following partition predicates:

- $\mathcal{C} \coloneqq \mathbb{F}$ is partitioned into singletons $\{x\}$. This is the *distinctness* predicate, namely the one that outputs 1 if and only if all the inputs $c_1, \ldots, c_r$ are distinct challenges. We implicitly assume that it is the default predicate, and we may omit it, i.e., we abbreviate a tree T supporting this predicate as an $\vec{n}$-tree of accepting transcripts
- $\mathcal{C} \coloneqq \mathbb{F}^m$, for some $m \in \mathbb{N}$, is partitioned into $\{(x, y) \mid y \in \mathbb{F}^{m-k}\}$ for all $x \in \mathbb{F}^k$. This is the *k-prefix distinctness* predicate, namely the one that outputs 1 if and only if all the inputs $c_1, \ldots, c_r$ have different prefixes of length k. We abbreviate this predicate into the number n of challenges as $n_{:k}$.
- $\mathcal{C} \coloneqq \mathbb{F}$ is partitioned into $\{x, -x\}$ for all x. We abbreviate this predicate into the number n of challenges as $n_{\pm}$.
- $\mathcal{C} \coloneqq \mathbb{F}^2$ is partitioned into $\{c \cdot x \mid c \in \mathbb{F}^*\}$ for all $x \in \{(0,0), (0,1)\} \cup \{(1, a) \mid a \in \mathbb{F}\}$ that captures the linear independence between two vectors. We abbreviate this predicate into the number n of challenges as n_{li} .

Definition 5.4.4 (ϵ-Uniform Partition)**.** *Let $\mathcal{C} \coloneqq \bigcup_{i=1}^{C} \mathcal{C}^{(i)}$. We say that $\{\mathcal{C}^{(i)}\}_{i \in [C]}$ is* ϵ-uniform *if there exists $\mathcal{I} \subseteq [C]$ such that $|\bigcup_{i \notin \mathcal{I}} \mathcal{C}^{(i)}| = \epsilon \cdot C$ and for all $i, j \in \mathcal{I}$: $|\mathcal{C}^{(i)}| = |\mathcal{C}^{(j)}|$.*

All the partitions defined in the above predicates satisfy this property. In particular, the distinctness predicate, the k-prefix distinctness (for all k) and the $n_{\pm}$ predicate use 0-uniform partitions, while n_{li} uses $1/(|\mathbb{F}| + 2)$-uniform partitions.

We now restate the guarantees of the (abstract) tree-builder of [ACK21, DG23].

Theorem 5.4.1 (Efficient Abstract Tree Builder)**.** *Consider any sets $\mathcal{S}_1, \ldots, \mathcal{S}_r$ that have an efficiently decidable partition $\mathcal{S}_i \coloneqq \bigcup_{j=1}^{C_i} \mathcal{S}_{i,j}$, and any $\vec{n} \coloneqq (n_1, \ldots, n_r) \in \mathbb{N}^r$ with $N \coloneqq \prod_{i=1}^{r} n_i$. There exists a probabilistic algorithm $\mathcal{T}$ such that for any Q-query abstract adversary $\mathcal{A}$ with success probability $\nu_{\mathcal{A}}$ against $(\mathcal{S}_1, \ldots, \mathcal{S}_r)$, $\mathcal{T}$ outputs an $\vec{n}$-abstract tree of transcript T with probability*

$$\nu_{\mathcal{T}} \geq \nu_{\mathcal{A}} - \frac{(Q+1)(\sum_{i=1}^{r} n_i - r)}{C}$$

where $C \coloneqq \min_{i \in [r]} C_i$.

Finally, we restate a theorem asserting the existence of an efficient tree-builder that can generate $(\vec{\phi}, \vec{n})$-trees of accepting transcripts, where $\vec{\phi}$ consists of partition predicates as defined above. Similarly to [DG23], our proof relies on the tree-builder constructed in the work of [ACK21], but we achieve a tighter bound since we do not incur in a quadratic dependence on the number of queries Q.

Theorem 5.4.2 (Efficient Tree Builder)**.** *Let Π be a $(2r+1)$-message public-coin interactive argument with challenge spaces $\mathcal{C}_1, \ldots, \mathcal{C}_r$. Consider any efficiently decidable and ϵ_i-uniform partition $\mathcal{C}_i \coloneqq \bigcup_{j=1}^{C_i} \mathcal{C}_{i,j}$, with $\epsilon_i \in \mathsf{negl}(\lambda)$ for all $i \in [r]$, with minimum partition size $C \coloneqq \min_i C_i$, and let $\vec{\phi} \coloneqq (\phi_1, \ldots, \phi_r)$ be the corresponding partition predicate. Consider the tree-building experiment in Fig. 5.7. There exists a probabilistic algorithm $\mathcal{A}$ such that for any $\vec{n} \coloneqq (n_1, \ldots, n_r) \in \mathbb{N}^r$, with $N \coloneqq \prod_{i=1}^{r} n_i$, and any (malicious) prover $\mathcal{P}^*$:*

$$\Pr\left[\mathsf{TB}^{\mathcal{A},\mathcal{P}^*}_{\Pi_{\mathsf{FS}},(\vec{\phi},\vec{n})}(\lambda)\right] \geq \Pr\left[\mathsf{KS}^{\mathcal{P}^*}_{0,\Pi_{\mathsf{FS}},\mathcal{R}}(\lambda)\right] - \frac{(Q+1)(\sum_{i=1}^{r} n_i - r)}{C} - Q \cdot \max_i \epsilon_i$$

Game $\mathsf{TB}^{\mathcal{A},\mathcal{P}^*}_{\Pi_{\mathsf{FS}},(\vec{\phi},\vec{n})}(\lambda)$

$\mathsf{pp} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda, \mathsf{pp}_{\mathbb{G}})$

$(\mathbb{x}, \pi) \leftarrow \mathcal{P}^{*\mathsf{H}}(\mathsf{pp})$

$\mathsf{T} \leftarrow \mathcal{A}^{\mathcal{P}^*}(\mathsf{pp}, \mathbb{x}, \pi)$

return $\mathcal{V}^{\mathsf{H}}(\mathsf{pp}, \mathbb{x}, \pi) = 1 \wedge \mathsf{IsAccepting}((\vec{\phi}, \vec{n}), \mathsf{pp}, \mathbb{x}, \mathsf{T})$

Figure 5.7: Tree-building security game. $\mathcal{A}$ is given black-box access to $\mathcal{P}^*$.

where $\mathcal{A}$ makes in expectation at most $(Q+1)(N-1)+1$ rewinding calls to $\mathcal{P}^*$, and Q is an upper bound to the number of RO queries of $\mathcal{P}^*$.

Proof. Without loss of generality, we assume that $\mathcal{P}^*$ is deterministic because if we can prove the theorem for every choice of $\mathcal{P}^*$'s randomness, then by averaging we also prove the theorem for arbitrary $\mathcal{P}^*$. Thus, the only source of randomness in the game $\mathsf{KS}^{\mathcal{P}^*}_{0,\Pi_{\mathsf{FS}},\mathcal{R}}(\lambda)$, and of the success probability of $\mathcal{P}^*$ is the choice of the random oracle H.

For all $i \in [r]$, let $\mathsf{H}_i \colon \{0,1\}^* \to [C_i]$. Moreover, for all $i \in [r]$, let $\mathcal{I}_r$ the subset of $[C_i]$ for which the challenge space $\mathcal{C}_i$ admits an ϵ_i-uniform partition.

We construct an abstract adversary $\mathcal{B}$ against the sets $[\mathcal{I}_1], \ldots, [\mathcal{I}_r]$, having access to random oracles $\mathsf{H}^* := (\mathsf{H}^*_1, \ldots, \mathsf{H}^*_r)$ and to the malicious prover $\mathcal{P}^*$. It does the following:

- Get $\mathsf{pp} \leftarrow \mathsf{Setup}(\mathsf{pp}_{\mathbb{G}})$ and run $\mathcal{P}^*$ on input pp
- When $\mathcal{P}^*$ makes an oracle query to H_i on input a message a, search through the (initially empty) table T for an entry of the form $(i, a, (\cdot, c))$, and return c. If no such query exists, query $\mathsf{H}^*_i(a)$ and obtain the value j
  - If $j \in \mathcal{C}_i$ then sample $c \leftarrow\!\!\$\ \mathcal{C}_{i,j}$ uniformly at random, add $(i, a, (j, c))$ to T, and return c as the answer to $\mathcal{P}^*$
  - Otherwise abort
- When $\mathcal{P}^*$ outputs $(\mathbb{x}, \pi := (a_1, \ldots, a_{r+1}))$, run $v \leftarrow \mathcal{V}^{\mathsf{H}}(\mathsf{pp}, \mathbb{x}, \pi)$, where H is determined by T, and output $(((\mathbb{x}, a_1), a_2, \ldots, a_{r+1}), v)$.

We now define our tree-builder algorithm $\mathcal{A}$. Given oracle access to $\mathcal{P}^*$, it emulates the abstract adversary $\mathcal{B}$, then run the abstract tree-builder $\mathcal{T}$ (cf. Theorem 5.4.1) on $\mathcal{B}$. If $\mathcal{T}$ returns an $\vec{n}$-abstract tree of transcripts T, then $\mathcal{A}$ returns a $(\vec{\phi}, \vec{n})$-tree of accepting transcripts T_Π for Π_{FS} as follows:

- For each vertex at depth $i \in [r+1]$ of T with label a_i, the same vertex for T_Π has label a_i too
- For each edge labeled j going from a vertex labeled a at depth $i \in [r]$, the same edge for T_Π has the label c, where c is the unique challenge such that $(i, a, (c, j)) \in T$

First, we observe that the abstract adversary $\mathcal{A}$ is nearly as efficient as $\mathcal{P}^*$ since it runs $\mathcal{P}^*$ once and does some other tasks in comparable time (managing the table T, running the

algorithm Setup and the verifier procedure $\mathcal{V}_{\mathsf{FS}}$). The tree-builder $\mathcal{A}$ invokes once on $\mathcal{B}$ the tree-builder $\mathcal{T}$ of [ACK21] (cf. Theorem 5.4.1), hence inheriting its expected running time: concretely, the expected running time of $\mathcal{A}$ is at most $(Q-1)\cdot(N+1)+1$ times the running time of $\mathcal{P}^*$.

We show that if $\mathcal{B}$ does not abort, T_Π is indeed a $(\vec{\phi},\vec{n})$-tree of accepting transcripts. It is clear that T_Π is of the right arity. For any vertex v at depth $i\in[r]$, we know that the edges coming from v are labeled with different $(j_{i,1},\ldots,j_{i,n_i})$ in T. This implies that for T_Π, the edges coming from the corresponding vertex v has challenges $(c_{i,1},\ldots,c_{i,n_i})$ satisfying $c_{i,k}\in\mathcal{C}_{i,j_{i,k}}$ for all $k\in[n_i]$. Hence, T_Π satisfies the partition predicate $\vec{\phi}$.

Moreover, $\mathcal{B}$ perfectly simulates the random oracles H for $\mathcal{P}^*$. For all $i\in[r]$ it first samples a partition index j and then samples from the j-th partition $\mathcal{C}_{i,j}$ a random challenge: this procedure is equivalent to uniformly sampling a challenge from the challenge space $\mathcal{C}_i$ since the partitions are ϵ_i-uniform and, in particular, have the same size. Then we have that the winning probability of $\mathcal{A}$ is the same as $\mathcal{P}^*$, conditioned on the event that $\mathcal{B}$ does not abort. We now bound the probability that $\mathcal{B}$ aborts. Since for all $i\in[r]$ the partition $\{\mathcal{C}_{i,j}\}_{j\in[C_i]}$ is ϵ_i-uniform, the probability that on input a message a the abstract adversary $\mathcal{B}$ aborts is at most $\epsilon_i\in\mathsf{negl}(\lambda)$. By union bound on the number of RO queries we derive that $\mathcal{B}$ aborts with probability at most $Q\cdot\max_i\epsilon_i$. □

5.5 Simulation extractability of Hyrax

5.5.1 An overview of Hyrax

We give a brief overview and provide some intuition on the multilinear polynomial commitment `HyraxPC` that was first introduced in [WTs+17]. It is a polynomial commitment scheme equipped with a $(\mu+1)$-rounds Eval protocol for a μ-variate multilinear polynomial.

Let $\mathsf{bin}\colon\mathbb{N}\to\{0,1\}^*$ be the function that computes the binary representation of an integer. Moreover, given a matrix $\vec{T}$ with n rows and m columns and a column vector $\vec{w}$ with $n\cdot m$ rows, we say that the *column-major order* of $\vec{T}$ is $\vec{w}$ if and only if $\forall i,j: T_{i,j}=w_{i+n\cdot(j-1)}$.

To evaluate on a point $x\in\mathbb{F}^\mu$ a multilinear polynomial $p(X_1,\ldots,X_\mu)$, given its evaluations $(w_i)_{i\in[2^\mu]}$ over the hypercube $\{0,1\}^\mu$, we can use the following formula to compute $p(x)$:

$$\begin{aligned}\sum_{k\in\{0,1\}^\mu} p(k)\cdot\prod_{i=1}^{\mu}\widetilde{eq}(x_i,k_i) &= \sum_{k\in[2^\mu]} w_k\cdot\prod_{i=1}^{\mu}\widetilde{eq}(x_i,\mathsf{bin}(k)_i)\\ &= \sum_{k\in[2^{\mu/2}]}\sum_{\ell\in[2^{\mu/2}]} w_{k+2^{\mu/2}\ell}\cdot\underbrace{\prod_{i=1}^{\mu/2}\widetilde{eq}(x_i,\mathsf{bin}(k)_i)}_{\vec{L}}\cdot\underbrace{\prod_{i=1}^{\mu/2}\widetilde{eq}(x_{\mu/2+i},\mathsf{bin}(\ell)_i)}_{\vec{R}}\\ &= \vec{L}\cdot\vec{T}\cdot\vec{R}^\top\end{aligned}$$

where $\vec{T}$ is the $2^{\mu/2}\times 2^{\mu/2}$ matrix whose column-major order is $(w_i)_{i\in 2^\mu}$, i.e., $\forall i,j\in[2^\mu/2]: T_{i,j}\coloneqq w_{i+2^{\mu/2}\cdot(j-1)}$.

The prover $\mathcal{P}$ commits individually to each row of $\vec{T}$, using `Pedersen`, and outputs a list of commitments $\vec{C}\coloneqq(C_1,\ldots,C_{2^{\mu/2}})$. We observe that the verifier $\mathcal{V}$ can compute a commitment to $\vec{L}\cdot\vec{T}$, namely $C_{\vec{L}\cdot\vec{T}}\leftarrow\prod_{k=1}^{2^{\mu/2}}C_k^{L_k}$ since this just requires public information.

- $\mathsf{Setup}(\mu, \mathsf{pp}_{\mathbb{G}})$: abort if μ is odd. Parse $\mathsf{pp}_{\mathbb{G}}$ as a group description $(\mathbb{G}, \mathbb{F})$. Sample $g_1, \ldots, g_{\mu/2}, h \leftarrow\!\!\$\ \mathbb{G}$ and output $\mathsf{pp} = (\mathbb{F}, \mathbb{G}, g, g_1, \ldots, g_{\mu/2}, h)$.
- $\mathsf{Com}(\mathsf{pp}, p(X_1, \ldots, X_\mu); \boldsymbol{\omega})$: $\forall i \in [2^\mu]$ let $w_i := p(\mathsf{bin}(i))$, define $\vec{T} \in \mathbb{F}^{2^{\mu/2} \times 2^{\mu/2}}$ s.t. $\forall i, j \in [2^{\mu/2}] : T_{i,j} := w_{i+2^{\mu/2}\cdot(j-1)}$.
  (Namely, $\vec{w}$ is the column-major order of $\vec{T}$.)
  $\forall i \in [2^{\mu/2}]$, sample $\omega_i \leftarrow\!\!\$\ \mathbb{F}$ and compute $C_i := \prod_{j=1}^{2^{\mu/2}} g_j^{T_{i,j}} \cdot h^{\omega_i}$.
  Output $\vec{C} := (C_1, \ldots, C_{2^{\mu/2}})$ and opening $\boldsymbol{\omega} := (\omega_i)_{i \in [2^{\mu/2}]}$.
- $\mathsf{Eval}\langle \mathcal{P}(p, \boldsymbol{\omega}, v, \omega_v), \mathcal{V} \rangle(\mathsf{pp}, \mathbf{C}, x, C_v)$: given a commitment C_v as public input, with an evaluation point $x \in \mathbb{F}^\mu$
  1. Let $\widetilde{eq}_L(Y) = \prod_{i=1}^{\mu/2} \widetilde{eq}(x_i, Y_i)$ and $\widetilde{eq}_R(Y) = \prod_{i=\mu/2+1}^{\mu} \widetilde{eq}(x_i, Y_i)$.
  2. $\mathcal{P}$ and $\mathcal{V}$ compute $P = C_v \cdot \prod_{k=1}^{2^{\mu/2}} C_k^{\widetilde{eq}_L(\mathsf{bin}(k))}$ and $\vec{r} = (\widetilde{eq}_R(k))_{k \in \{0,1\}^{\mu/2}}$.
  3. $\mathcal{P}$ also computes $\omega_P := \omega_v + \sum_{k \in [2^{\mu/2}]} \omega_k \cdot \widetilde{eq}_L(\mathsf{bin}(k))$ and
  $$\vec{l} := \left(\sum_{k \in [2^{\mu/2}]} T_{k,j} \cdot \widetilde{eq}_L(\mathsf{bin}(k)) \right)_{j \in [2^{\mu/2}]}$$
  4. $\mathcal{P}$ and $\mathcal{V}$ engage in `LogDotProd`, on input $((2^{\mu/2}, g, \mathbf{g}, h), (P, \mathbf{r}), (\mathbf{l}, v, \omega_P))$, to prove that $P = g^v \cdot \vec{g}^{\vec{l}} \cdot h^{\omega_P}$ and $v = \langle \vec{l}, \vec{r} \rangle$.
- $\mathsf{Open}\langle \mathcal{P}(p, \boldsymbol{\omega}), \mathcal{V} \rangle(\mathsf{pp}, \mathbf{C})$:
  1. $\mathcal{V}$ samples challenge $x \leftarrow\!\!\$\ \mathbb{F}^\mu$, $\mathcal{P}$ replies with $C_v = g^v h^{\omega_v}$, where $v = p(x)$
  2. $\mathcal{P}$ and $\mathcal{V}$ engage in Eval on input $(\mathsf{pp}, \mathbf{C}, x, C_v)$.

Figure 5.8: Description of `HyraxPC`. The function $\mathsf{bin}\colon \mathbb{N} \to \{0,1\}^*$ computes the binary representation of an integer. The protocol `LogDotProd` is defined in Fig. 5.9.

Finally, $\mathcal{P}$ and $\mathcal{V}$ can run an inner product argument to confirm that $(\vec{L} \cdot \vec{T}) \cdot \vec{R}^\top$ equals $p(x)$, supposedly committed in C_v, having access to the commitments $C_{\vec{L}\cdot\vec{T}}$ and $C_{\vec{R}}$. This part is handled by a logarithmic-size dot product proof `LogDotProd` (see Fig. 5.9) that is similar to the inner product argument of Bulletproofs [BBB+18], but also achieves zero-knowledge.

5.5.2 Proof of the simulation extractability of Hyrax

We recall that `HyraxPC` has been proved $(2^{\mu/2}, (4_\pm)^{\mu/2}, 2)$ computational special sound in [DG23].[7]

[7] Their proof has some technical flaw, but we show how to fix it.

1. Set $n_0 \leftarrow n, \mathbf{g}^{(0)} \leftarrow \mathbf{g}, P^{(0)} \leftarrow P, \mathbf{a}^{(0)} \leftarrow \mathbf{a}, \mathbf{x}^{(0)} \leftarrow \mathbf{x}, y^{(0)} \leftarrow y, r_P^{(0)} \leftarrow r_P$.
   For $i = 1, \ldots, k$:
   (a) $L_i \leftarrow g^{y_L^{(i)}} \cdot \left(\mathbf{g}^{(i-1)}\right)^{\mathbf{x}_{[:n_i]}^{(i-1)}} \cdot h^{r_L^{(i)}}, R_i \leftarrow g^{y_R^{(i)}} \cdot \left(\mathbf{g}^{(i-1)}\right)^{\mathbf{x}_{[n_i:]}^{(i-1)}} \cdot h^{r_R^{(i)}}$.
   (b) $\mathcal{V}$ sends challenge $c_i \xleftarrow{\$} \mathbb{F}$.
   (c) $\mathcal{P}$ and $\mathcal{V}$ both compute $P^{(i)} \leftarrow L_i^{c_i^2} \cdot P^{(i-1)} \cdot R_i^{c_i^{-2}}$ and
   $$\mathbf{a}^{(i)} \leftarrow c_i^{-1} \cdot \mathbf{a}_{[:n_i]}^{(i-1)} + c_i \cdot \mathbf{a}_{[n_i:]}^{(i-1)}, \quad \mathbf{g}^{(i)} \leftarrow \left(\mathbf{g}_{[:n_i]}^{(i-1)}\right)^{c_i^{-1}} \circ \left(\mathbf{g}_{[n_i:]}^{(i-1)}\right)^{c_i}.$$
   (d) $\mathcal{P}$ computes $\mathbf{x}^{(i)} \leftarrow c_i \cdot \mathbf{x}_{[:n_i]}^{(i-1)} + c_i^{-1} \cdot \mathbf{x}_{[n_i:]}^{(i-1)}$ and
   $$y^{(i)} \leftarrow c_i^2 \cdot y_L^{(i)} + y^{(i-1)} + c_i^{-2} \cdot y_R^{(i)}, \quad r_P^{(i)} \leftarrow c_i^2 \cdot r_L^{(i)} + r_P^{(i-1)} + c_i^{-2} \cdot r_R^{(i)}.$$
2. Set $\hat{g} \leftarrow \mathbf{g}^{(k)}, \hat{P} \leftarrow P^{(k)}, \hat{a} \leftarrow \mathbf{a}^{(k)}, \hat{x} \leftarrow \mathbf{x}^{(k)}, \hat{y} \leftarrow y^{(k)}, \hat{r_P} \leftarrow r_P^{(k)}$. $\mathcal{P}$ samples $d, r_\beta, r_\delta \xleftarrow{\$}$ and sends $\beta \leftarrow g^d \cdot h^{r_\beta}, \delta \leftarrow \hat{g}^d \cdot h^{r_\delta}$.
3. $\mathcal{V}$ sends challenge $c \xleftarrow{\$} \mathbb{F}$.
4. $\mathcal{P}$ sends $z_1 \leftarrow d + c \cdot \hat{y}$ and $z_2 \leftarrow \hat{a} \cdot (c \cdot \hat{r_P} + r_\beta) + r_\delta$.
5. $\mathcal{V}$ checks that $\left(\hat{P}^c \cdot \beta\right)^{\hat{a}} \cdot \delta \stackrel{?}{=} \left(\hat{g} \cdot g^{\hat{a}}\right)^{z_1} \cdot h^{z_2}$.

Figure 5.9: Description of `LogDotProd`. For $n = 2^k$, $\mathcal{R}_{\texttt{LogDotProd}} = \{((n, g, \mathbf{g}, h), (P, \mathbf{a}), \mathbf{x}, y, r_P) : P = g^y \cdot \mathbf{g}^{\mathbf{x}} \cdot h^{r_P}, y = \langle \mathbf{x}, \mathbf{a} \rangle\}$.

The protocol Eval is a public coin interactive argument for the relation:

$$\mathcal{R}_{\mathsf{Eval}} = \left\{ \mathsf{ck}, (C_p, x, C_v), (p, \omega_p, v, \omega_v) : \begin{array}{l} C_p = \mathsf{Com}(\mathsf{ck}, p; \omega_p), \\ C_v = \mathsf{Com}(\mathsf{ck}, v; \omega_v), \\ p(x) = v \wedge p \text{ is multilinear}, \end{array} \right\}$$

In [DG23], the authors prove the computational special soundness of Eval under the additional condition that the evaluation point $x \in \mathbb{F}^\mu$ is sampled uniformly at random. Although conceptually sound, their statement does not fulfill the formalism of Definition 5.3.2. We fix this inconsistency of the notation of Dao and Grubbs by, first, defining a different relation:

$$\mathcal{R}_{\mathsf{Open}} = \{\mathsf{ck}, (C), (p, \omega) : C = \mathsf{Com}(\mathsf{ck}, p; \omega) \wedge p \text{ is multilinear}\}$$

We then define a protocol Open which, basically, runs Eval on a random challenge x (cf. Fig. 5.8), and we prove computational special soundness for Open: crucially, in the proof we can rewind the prover feeding different challenges x.

By additionally proving that the Open protocol achieves μ-zero-knowledge and μ-unique-response, we derive that HyraxPC is simulation-extractable.

We start by proving that the Eval protocol of HyraxPC achieves μ-ZK and μ-UR, where μ is the number of variables of the polynomials committed.

Lemma 5.5.1. HyraxPC.Eval *is μ-ZK and μ-UR.*

Proof. We start by proving the μ-ZK property.

The simulator $\mathcal{S}_{\texttt{HyraxPC}}$ for HyraxPC.Eval, on input $(\mathsf{pp}, \vec{C}, x, C_v)$, computes the instance $(P, \vec{r})$ for LogDotProd, as the honest prover would do, and then invokes the simulator $\mathcal{S}_{\texttt{LogDotProd}}$ on input $((2^{\mu/2}, g, \vec{g}, h), (P, \vec{r}))$ that does the following:

1. For $i \in [\mu]$ samples the group elements L_i and R_i at random. Retrieves the challenge c_i (as the honest prover would do) and computes the values $P^{(i)}, \vec{a}^{(i)}$ and $\vec{g}^{(i)}$ accordingly.
2. Let $\hat{g} := \vec{g}^{(\mu)}$, $\hat{P} := P^{(\mu)}$ and $\hat{a} := \vec{a}^{(\mu)}$. Samples random field elements c, z_1, z_2 and a random group element β. Finally, computes $\delta := (\hat{g} \cdot g^{\hat{a}}) \cdot h^{z_2} / (\hat{P}^c \cdot \beta)^{\hat{a}}$ and invokes RePro to make c be the final challenge output by $\mathcal{V}$ on input the transcript, including β and δ.

$\mathcal{S}_{\texttt{HyraxPC}}$ only makes a single RO reprogramming, in particular when invokes the simulator for LogDotProd. The output of $\mathcal{S}_{\texttt{HyraxPC}}$ is indistinguishable from that of a real transcript: the random group elements L_i, R_i are indistinguishable from the hiding commitments used in a real proof; similarly, the distribution of the field elements z_1, z_2 and the elements β, δ is also indistinguishable from the one in a real proof.

As for the μ-UR property, it is sufficient to notice that once the transcript of a proof has been fixed up to μ-th round, if we are given two accepting last-round pairs $(z_1, z_2) \neq (z_1', z_2')$ we can always reduce to the discrete log problem as we can find a non-trivial relation between the generators g and h. □

We observe that this result implies the following corollary.

Corollary 5.5.1. HyraxPC.Open *is $(\mu+1)$-ZK and $(\mu+1)$-UR.*

Proof. Since the Open protocol consists of a random challenge sent by the verifier followed by an execution of the protocol Eval, the proof of $(\mu+1)$-UR follows directly by the μ-UR of Eval. Finally, it is easy to see that we can define a simple simulator that first obtains the random coin of the verifier and then runs the code of the simulator $\mathcal{S}_{\texttt{HyraxPC}}$ defined above that needs to reprogram the random oracle only at the μ-th round and produces transcripts indistinguishable from those of real proofs. □

Finally, we show that HyraxPC.Open is computational special sound. Despite similar to the proof of special soundness of [DG23], we notice that we rely on the prefix distinctness predicate to extract the witness.

Lemma 5.5.2. *For all $\mu \in \mathbb{N}$, the protocol* HyraxPC.Open *(cf. Fig. 5.8) is computational special sound, i.e., there exist a tree extractor $\mathcal{TE}_{\texttt{HyraxPC}}$ and an EPT adversary $\mathcal{B}$ such that given an $\vec{n} := ((2^{\mu/2})_{:\mu/2}, (4_{\pm})^{\mu/2}, 2)$-tree of accepting transcripts (produced by an adversary $\mathcal{A}$) for the $(\mu+2)$-rounds* Open *protocol, we have that:*

$$\mathbf{Adv}^{\mathsf{SS}}_{\mathsf{Open},\vec{n}}(\mathcal{TE}_{\mathsf{Open}}, \mathcal{A}) \leq 2^{\mu/2} \left(\mathbf{Adv}^{\mathsf{DL}}_{\mathbb{G}}(\mathcal{B}) + \frac{1}{|\mathbb{F}|} \right)$$

1. $P \to V$: The prover sends 3α different $(\log m)$-variate multilinear polynomials $E_1, \ldots, E_\alpha$, $\mathrm{dim}_1, \ldots, \mathrm{dim}_\alpha$, $\mathrm{read\_ts}_1, \ldots, \mathrm{read\_ts}_\alpha$ and α different $(\frac{1}{c}\log N)$-variate multilinear polynomials $\mathrm{final\_ts}_1, \ldots, \mathrm{final\_ts}_\alpha$, where $\forall i \in [\alpha]$: E_i is purported to specify the values of each of the m reads into T_i, dim_i is the multilinear extension of nz_i, while $\mathrm{read\_ts}_i$ and $\mathrm{final\_ts}_i$ are the "counter polynomials" for the i-th sub-table T_i.
2. $V \to P$: The verifier picks a random $r \in \mathbb{F}^{\log m}$ and sends it to $\mathcal{P}$. The verifier makes one evaluation query to $\widetilde{a}$ to learn $v := \widetilde{a}(r)$.
3. $P \leftrightarrow V$: sum-check protocol to check that $v = \sum_{k \in \{0,1\}^{\log m}} \widetilde{eq}(r, k) \cdot \hat{f}(E_1(k), \ldots, E_\alpha(k))$
   - $\log m$ rounds of interaction in which the prover sends univariate polynomials and the verifier replies with a random coin
   - The verifier checks that $E_i(r_z) = v_i$ for all $i \in [\alpha]$, where $(v_i)_i$ are values provided by the prover at the end of the sum-check protocol. The verifier checks the equation above with one oracle query to each E_i.
4. $V \to P$: The verifier picks two random field elements γ, τ
5. $P \leftrightarrow V$: α sum-check-based protocols (in parallel) for "grand products" to check that $H_{\tau,\gamma}(WS) = H_{\tau,\gamma}(RS) \cdot H_{\tau,\gamma}(S)$. The verifier checks the equations hold with an oracle query to each of $E_i, \mathrm{dim}_i, \mathrm{read\_ts}_i, \mathrm{final\_ts}_i$.

Figure 5.10: A description of Lasso [STW24]. T is a decomposable lookup table of size N.

Proof. The first layer in the tree of transcripts consists of $2^{\mu/2}$ distinct verifier's challenges $((x_{i,\ell})_{\ell \in [\mu]})_{i \in [2^{\mu/2}]}$, each one corresponding to an evaluation point; the rest of the tree then corresponds to an instance of $\texttt{LogDotProd}$. The tree extractor $\mathcal{TE}_{\texttt{HyraxPC}}$ runs the tree extractor $\mathcal{TE}_{\texttt{LogDotProd}}$ (that is similar to the one of the inner-product argument of Bulletproofs, and so we refer to [BBB+18, DG23]) on each $((4^{\mu/2})_{\pm}, 2)$-subtree to recover the underlying linear combinations

$$\vec{l}^{(i)} := \left(\sum_{k \in [2^{\mu/2}]} w_{k+2^{\mu/2}(j-1)} \cdot \widetilde{eq}((x_{i,1}, \ldots, x_{i,\mu/2}), \mathsf{bin}(k)) \right)_{j \in [2^{\mu/2}]}$$

Here, the tree extractor $\mathcal{TE}_{\texttt{LogDotProd}}$ either succeeds or we can build an adversary $\mathcal{B}$ against the discrete log problem in $\mathbb{G}$.

Finally, for each $j \in [2^{\mu/2}]$, we can use the j-th entry of all the $\vec{l}^{(i)}$, for $i \in [2^{\mu/2}]$, corresponding to the $2^{\mu/2}$ different verifier's challenges, and we solve for $w_{k+2^{\mu/2}(j-1)}$ for all $k \in [2^{\mu/2}]$. This is possible because the Lagrange polynomials $\{\widetilde{eq}((x_{i,1}, \ldots, x_{i,\mu/2}), k)\}_{i,k}$ are independent since the challenges $((x_{i,\ell})_{\ell \in [\mu]})_{i \in [2^{\mu/2}]}$ crucially satisfy the $\mu/2$-prefix distinctness predicate $\phi_{:\mu/2}$. □

5.6 Simulation extractability of Lasso

In this section, we recall the Lasso indexed lookup argument [STW24], and we show how we can apply Theorem 5.3.1 to prove that a zero-knowledge version of Lasso is simulation-extractable.

5.6.1 Overview of Lasso

The starting point of Lasso is to model the lookup argument in a sparse way, as it is done in schemes like Caulk [ZBK+22] or Baloo [ZGK+22]: given a commitment to a table $t \in \mathbb{F}^n$ and a commitment to a vector $a \in \mathbb{F}^m$, the prover can prove to know a *sparse* matrix $M \in \mathbb{F}^{m \times n}$ such that (1) each row of M is a unit vector, i.e., there are $n-1$ zeroes and one cell is equal to 1, and (2) $M \cdot t = a$. This turns out to be equivalent, up to negligible soundness error $\log m \cdot |\mathbb{F}|^{-1}$, to check that:

$$\sum_{y \in \{0,1\}^{\log n}} \widetilde{M}(r, y) \cdot \widetilde{t}(y) = \widetilde{a}(r) \tag{5.1}$$

when $r \in \mathbb{F}^{\log m}$ is chosen uniformly at random by the verifier after the prover has sent (a commitment to) $\widetilde{M}$. The core idea of Lasso is to use Surge, a generalization of the Spark commitment scheme [Set20], to commit to $\widetilde{M}$ and then prove that Eq. (5.1) holds by evaluating $\widetilde{M}$ in a point (r, r_x) chosen by the verifier. To do that, the table t needs to be "decomposable" as we define hereafter.[8]

Definition 5.6.1 (Decomposable Table)**.** *A table $t \in \mathbb{F}^n$ is* decomposable *if there is $k \geq 1$ and $\alpha := kc$ tables $t_1, \ldots, t_\alpha$, each of size $n^{1/c}$, as well as an α-variate multilinear polynomial $\hat{f}$ such that for every $(r_1, \ldots, r_c) \in (\{0,1\}^{\frac{1}{c}\log n})^c$:*

$$t[r_1, \ldots, r_c] = \hat{f}(t_1[r_1], \ldots, t_k[r_1], t_{k+1}[r_2], \ldots, t_{2k}[r_2], \ldots, t_{\alpha-k+1}[r-c], \ldots, t_\alpha[r_c])$$

Let $\mathrm{nz}(i)$ denote the (unique) column in the i-th row of M that contains the value 1. First, we observe that can rewrite Eq. (5.1) as:

$$\sum_{k \in \{0,1\}^{\log m}} \widetilde{eq}(k, r) \cdot t[\mathrm{nz}(i)] = \widetilde{a}(r) \tag{5.2}$$

and if t is decomposable we can further rewrite it as:

$$\sum_{k \in \{0,1\}^{\log m}} \widetilde{eq}(k, r) \cdot \hat{f}(t_1[\mathrm{nz}_1(i)], t_k[\mathrm{nz}_1(i)], \ldots, t_{a-k+1}[\mathrm{nz}_c(i)], \ldots, t_\alpha[\mathrm{nz}_c(i)]) = \widetilde{a}(r) \tag{5.3}$$

for some polynomial $\hat{f}$, where $\mathrm{nz}_1(i), \ldots \mathrm{nz}_c(i)$ are the "chunks" in which $\mathrm{nz}(i)$ has been decomposed.

For all $j \in [c]$, let $\mathrm{dim}_j \colon \mathbb{F}^{\log m} \to \mathbb{F}$ be equal to $\widetilde{\mathrm{nz}}_j$. Moreover, for all $i \in [\alpha]$, let $E_i \colon \mathbb{F}^{\log m} \to \mathbb{F}$ be the $\log m$-variate multilinear polynomial that interpolates all the m lookups into t_i, namely $\forall k \in \{0,1\}^{\log m}$, we have that $E_i(k) := t_i[\mathrm{dim}_i(k)]$. Given this, we can rewrite Eq. (5.3) simply as:

$$\sum_{k \in \{0,1\}^{\log m}} \widetilde{eq}(k, r) \cdot \hat{f}(E_1(k), \ldots, E_\alpha(k)) = \widetilde{a}(r) \tag{5.4}$$

In Lasso, the prover commits to M sending commitments to $\mathrm{dim}_1, \ldots, \mathrm{dim}_c$, $E_i, \ldots, E_\alpha$ and the "counter polynomials" for the i-th sub-table T_i, $\mathrm{read\_ts}_i$ and $\mathrm{final\_ts}_i$. Then, the prover and the verifier engage in a sum-check protocol to check that Eq. (5.4) holds. Finally, the prover needs to convince the verifier that the polynomials E_i are actually encoding the values read from the (honest) memory t_i: to do that, they apply a memory checking procedure [BEG+91]

[8] In previous work, this is also referred to as *Spark-only structure* (SOS).

Setup Phase. Let $\mathtt{pp} \coloneqq (\mathtt{pp}_{\mathbb{G}}, g, g_1, \ldots, g_{\mu/2}, h)$, where $\mu \coloneqq \max(\mathtt{deg}(\hat{f}), \log m, \frac{1}{c}\log N)$, $\mathtt{deg}(\hat{f})$ is the total degree of g and $(g_1, \ldots, g_{\mu/2})$ are random generators of $\mathbb{G}$. Let $\mathtt{pp}_{\mathtt{Pedersen},1} \coloneqq (\mathtt{pp}_{\mathbb{G}}, g, h)$ be the parameters for `Pedersen` with message space $\mathbb{F}$; for $\nu > 1$, let $\mathtt{pp}_{\mathtt{Pedersen},\nu} \coloneqq (\mathtt{pp}_{\mathbb{G}}, g_1, \ldots, g_\nu, h)$ be the parameters for `Pedersen` with message space $\mathbb{F}^\nu$. For $\nu \in \mathbb{N}$ let $\mathtt{pp}_{\mathtt{HyraxPC},2\nu} \coloneqq (\mathtt{pp}_{\mathbb{G}}, g_1, \ldots, g_\nu)$ be the parameters for `HyraxPC` with message space $\mathbb{F}[X_1, \ldots, X_{2\nu}]$.

Interaction Phase.

1. $\mathcal{P}$ sends to $\mathcal{V}$ `HyraxPC` commitments to 3α different $(\log m)$-variate multilinear polynomials $E_1, \ldots, E_\alpha$, $\mathrm{dim}_1, \ldots, \mathrm{dim}_\alpha$, $\mathrm{read\_ts}_1, \ldots, \mathrm{read\_ts}_\alpha$ and α different $(\frac{1}{c}\log N)$-variate multilinear polynomials $\mathrm{final\_ts}_1, \ldots, \mathrm{final\_ts}_\alpha$, where $\forall i \in [\alpha]$: E_i is purported to specify the values of each of the m reads into T_i, dim_i is the multilinear extension of nz_i, while $\mathrm{read\_ts}_i$ and $\mathrm{final\_ts}_i$ are the "counter polynomials" for the i-th sub-table T_i.
2. $\mathcal{V}$ picks a random $r \in \mathbb{F}^{\log m}$ and sends it to $\mathcal{P}$.
3. $\mathcal{P}$ sends a `Pedersen` commitment C_v to the value v supposedly equal to $\widetilde{a}(r)$.
4. $\mathcal{P}$ and $\mathcal{V}$ engage in a sum-check to check that $v = \sum_{k \in \{0,1\}^{\log m}} u(k)$, where $u(X) \coloneqq \widetilde{eq}(r, X) \cdot \hat{f}(E_1(X), \ldots, E_\alpha(X))$: after $\log m$ rounds of interaction, the prover sends a `Pedersen` commitment C_{e_x} to the value e_x supposedly equal to $u(r_z)$
5. $\mathcal{P}$ sends the `Pedersen` commitments $C_{v_1}, \ldots, C_{v_\alpha}$ to values $v_1, \ldots, v_\alpha$, supposedly equal to $E_1(r_z), \ldots, E_\alpha(r_z)$
6. $\mathcal{P}$ and $\mathcal{V}$ engage in $\mathtt{GenPf}_{g,\alpha}$ to check that $\hat{f}(v_1, \ldots, v_\alpha) = e_x \widetilde{eq}(r, r_z)^{-1}$
7. The verifier checks using `HyraxPC.Eval` that $E_i(r_z) = v_i$ for all $i \in [\alpha]$.
8. $\mathcal{V}$ picks two random field elements γ, τ.
9. For $i = 1$ to α:
   - $\mathcal{P}$ sends to $\mathcal{V}$ the `Pedersen` commitment C_{h_i} to the value h_i supposedly equal to $H_{\gamma,\tau}(WS_i)$ and $H_{\gamma,\tau}(RS_i) \cdot H_{\gamma,\tau}(S_i)$.
   - $\mathcal{P}$ and $\mathcal{V}$ engage in `GrandProd` to check that $H_{\tau,\gamma}(WS_i) = h_i$ and $H_{\tau,\gamma}(RS_i) \cdot H_{\tau,\gamma}(S) = h_i$.
10. $\mathcal{P}$ and $\mathcal{V}$ engage in `HyraxPC.Eval` to check that $v = \widetilde{a}(r)$

Figure 5.11: The indexed lookup argument zkLasso.

that finally results into a sum-check-based grand product argument. More in detail, let WS and RS be two sets accounting for the write and read operations, respectively, and let S be the final state of the memory. Every time a read operation (i.e., a lookup) is issued, a write operation is performed too with the goal of updating the "counter" (i.e., the timestamp) associated with that memory location. The goal of the prover is to convince the verifier that the invariant "every value that has been read must have been written" is maintained at the end of the lookup process, i.e., $WS = RS \cup S$. Lasso is not zero-knowledge since a proof essentially leaks evaluations of $\widetilde{M}$ in some random coins sent by the verifier.

5.6.2 Zero-Knowledge Lasso

We define the main protocol in Fig. 5.11. It uses `Pedersen`, `HyraxPC`, three (2-perfect special sound) Σ-protocols (see also Fig. 5.12) sharing the same setup:

1. $\mathcal{P}$ sends to $\mathcal{V}$: $\alpha \leftarrow g^{b_1} \cdot h^{b_2}, \beta \leftarrow g^{b_3} \cdot h^{b_4}, \gamma \leftarrow X^{b_3} \cdot h^{b_5}$, where $(b_1, \ldots, b_5) \leftarrow\$ \mathbb{F}^5$
2. $\mathcal{V}$ responds with challenge $c \leftarrow\$ \mathbb{F} \setminus \{0\}$
3. $\mathcal{P}$ sends to $\mathcal{V}$: $z_1 \leftarrow b_1 + cx$, $z_2 \leftarrow b_2 + cr_x$, $z_3 \leftarrow b_3 + cy$, $z_4 \leftarrow b_4 + cr_y$, $z_5 \leftarrow b_5 + c(r_z - r_x y)$

$\mathcal{V}$ checks that $\alpha \cdot C_x^c = g^{z_1} \cdot h^{z_2}$, $\beta \cdot C_y^c = g^{z_3} \cdot h^{z_4}$ and $\delta \cdot C_z^c = C_x^{z_3} \cdot h^{z_5}$

Figure 5.12: The Σ-protocol `ProdPf` to check that the prover knows (x, y, r_x, r_y, r_z) such that $C_x = g^x h^{r_x}$, $C_y = g^y h^{r_y}$, $C_z = g^{xy} h^{r_z}$, given the commitments (C_x, C_y, C_z) and generators g, h.

- `ProdPf` to prove that three commitments C_x, C_y, C_z satisfy $xy = z$,
- `DotProdPf` to prove that a multi-commitment $C_{\vec{x}}$ and a commitment C_y satisfy $y = \langle \vec{x}, \vec{a} \rangle$ for a public vector $\vec{a}$
- $\texttt{GenPf}_{\hat{f},n}$ to prove that n commitments $(C_{v_i})_i$ satisfy $\hat{f}((v_i)_{i \in [n-1]}) = v_n$

and the following sub-protocols:

- A protocol `SumCheck` (see Fig. 5.13) to reduce the task of proving that $\sum_{x \in \{0,1\}^\mu} p(x) = v$, given the commitments (C_p, C_v), to the claim that $p(r_x) = e_x$ for a random $r_x \in \mathbb{F}^\mu$ sampled randomly by the verifier, and some claimed value $e_x \in \mathbb{F}$, where C_{e_x} is provided by the prover at the end of the procedure.
- A sum-check-based protocol `GrandProd` for "grand products" (see Fig. 5.14).

On the instantiation of `GenPf` and `GrandProd`. If $\hat{f}$ is a simple string concatenation, we can exploit the homomorphism of `Pedersen` and reduce `GenPf` to a single invocation of a Σ-protocol for the equality of two commitments (cf. `EqPf` in [DG23, WTs+17]). As for `GrandProd`, we use a commit-and-prove version of the Thaler's grand product argument [Tha13] that is an optimized application of the GKR protocol for circuit evaluation to a circuit computing a binary tree of multiplication gates. Another possibility would be to use the protocol due to Setty and Lee [SL20] that reduces the communication cost, and hence the proof size, at the cost of committing to additional field elements.

Lemma 5.6.1 ([STW24])**.** *Lasso has soundness error* $O(\frac{m + \log m}{|\mathbb{F}|})$.

Lemma 5.6.2. *For all* $\Pi \in \{\texttt{ProdPf}, \texttt{DotProdPf}\}$, Π *is 2-perfect special sound, i.e., there exists a tree-extraction algorithm that can extract a valid witness for* Π *given any* 2*-tree of accepting transcripts.*

We analyze the computational special soundness of the sumcheck (sub)protocol in Fig. 5.13. Although very similar, our scheme is different from the one used in [DG23] since we change the way the prover computes the vector $\vec{a}$ of the batched evaluations. Besides a negligible improvement in the efficiency, this change allows us to provide a (tighter) extractor that relies only on the *distinctness* predicate.

Let $e_0 = v$. For $i = 1$ to μ:

1. $\mathcal{P}$ computes the polynomial $p_i(X) \coloneqq \sum_{x\in\{0,1\}^{\mu-i}} p(r_1, \ldots, r_{i-1}, X, x)$, parses it as a vector of coefficients, then sends $C_{p_i} \leftarrow \mathsf{Com}(\mathsf{pp}, p_i; \omega_{p_i})$ to $\mathcal{V}$.
2. $\mathcal{V}$ responds with challenge $r_i \leftarrow\!\!\$\ \mathbb{F}$.
3. $\mathcal{P}$ computes $e_i \leftarrow p_i(r_i)$, then sends $C_{e_i} \leftarrow \mathsf{Com}(\mathsf{pp}, e_i; \omega_{e_i})$ to $\mathcal{V}$.
4. $\mathcal{V}$ responds with challenges $w_i \leftarrow\!\!\$\ \mathbb{F}$.
5. $\mathcal{P}$ and $\mathcal{V}$ compute $\vec{a} \leftarrow (1, \ldots, 1, 2) + w_i\vec{r}_i^{\,k}$ and $C_{y_i} \leftarrow C_{e_{i-1}} C_{e_i}^{w_i}$. In addition, $\mathcal{P}$ computes $y_i \leftarrow e_{i-1} + w_i e_i$ and $\omega_{y_i} \leftarrow \omega_{e_{i-1}} + w_i\omega_{e_i}$.
6. $\mathcal{P}$ and $\mathcal{V}$ engage in `DotProdPf` on input $(\mathsf{pp}, (C_{p_i}, C_{y_i}, \vec{a}), (p_i, \omega_{p_i}, y_i, \omega_{y_i}))$.

It is left to check that $p(r_1, \ldots, r_\mu) = e_\mu$.

Figure 5.13: The protocol `SumCheck` to reduce the task of proving that $\sum_{x\in\{0,1\}^\mu} p(x) = v$, given the commitments (C_p, C_v), to the claim that $p(r_x) = e_\mu$ for a random $r_x \in \mathbb{F}^\mu$ sampled randomly by the verifier, and some claimed value $e_\mu \in \mathbb{F}$, where C_{e_μ} is provided by the prover at the end of the procedure.

Let $z_0 = r_1 = 0$. $\mathcal{P}$ also sets $e_1 \leftarrow v$. For $i = 1$ to $d - 1$:

1. If $i > 1$ $\mathcal{P}$ and $\mathcal{V}$ engage in a (i rounds) sum-check to reduce the task of proving that $\sum_{p\in\{0,1\}^i} g^{(i)}_{z_{i-1}}(p) = \widetilde{V}_i(z_{i-1})$ to the claim that $g^{(i)}_{z_{i-1}}(r_i) = e_i$, for some r_i and $C_{e_i} \leftarrow \mathsf{Com}(\mathsf{pp}, e_i; \omega_{e_i})$ provided by the prover by the end of the protocol.
2. $\mathcal{P}$ sends $C_{w_{1,i}} \leftarrow \mathsf{Com}(\mathsf{pp}, \widetilde{V}_{i+1}(r_i, 0); \omega_{w_{1,i}})$ and $C_{w_{2,i}} \leftarrow \mathsf{Com}(\mathsf{pp}, \widetilde{V}_{i+1}(r_i, 1); \omega_{w_{2,i}})$
3. $\mathcal{P}$ and $\mathcal{V}$ engage in `ProdPf` on input $(\mathsf{pp}, (C_{w_{1,i}}, C_{w_{2,i}}, C_{e_i}^{1/\widetilde{eq}(z_{i-1}, r_i)}), (w_{1,i}, \omega_{w_{1,i}}, w_{2,i}, \omega_{w_{2,i}}, e_i/\widetilde{eq}(z_{i-1}, r_i), \omega_{e_i}))$
4. $\mathcal{V}$ sends a challenge $\beta_i \leftarrow\!\!\$\ \mathbb{F}$
5. $\mathcal{P}$ and $\mathcal{V}$ set $z_i \leftarrow l_i(\beta_i)$, where $l_i(X)$ is the unique line such that $l_i(0) = (r_i, 0)$ and $l_i(1) = (r_i, 1)$
6. $\mathcal{P}$ and $\mathcal{V}$ set $C_{v_i} \leftarrow C_{w_{1,i}}^{(1-\beta_i)} \cdot C_{w_{2,i}}^{\beta_i}$. Additionally, $\mathcal{P}$ sets $v_i \leftarrow w_{1,i}(1 - \beta_i) + w_{i,2}\beta_i$

Finally, $\mathcal{P}$ and $\mathcal{V}$ engage in `HyraxPC.Eval` to prove that $\tilde{V}_d(z_{d-1}) = v_{d-1}$.

Figure 5.14: The protocol `GrandProd` to prove that the product of 2^d inputs equals v, given a commitment C_v and a commitment to the MLE $\widetilde{V}_d$ of the input vector to a binary-tree circuit of depth d. The output gate is labelled 0, and the two inputs to a layer-i gate labelled $p \in \{0, 1\}^i$ are labelled as $(p, 0)$ and $(p, 1)$ respectively; hence `GrandProd` allows to prove that $V_1(0) = v$. For all $i \in [d - 1]$ and for all $p \in \{0, 1\}^i$, we have that $g^{(i)}_z(p) \coloneqq \widetilde{eq}(z, p) \cdot \widetilde{V}_{i+1}(p, 0) \cdot \widetilde{V}_{i+1}(p, 1)$

On input $(C_a, T_1, \ldots, T_\alpha)$, the simulator does the following:

1. Sample a "dummy" witness $M \in \{0,1\}^{m \times n}$, such that all the rows of M are unit vectors.
2. Compute the vector of looked-up values $b \leftarrow M \cdot T$
3. Run Lasso prover, until the second to last round, on input $(C_b, T_1, \ldots, T_\alpha)$ and witness M, where $C_b \leftarrow\!\$\ \mathsf{Com}(\mathsf{pp}, \widetilde{b})$
4. To prove that $v = \widetilde{a}(r)$, use the ZK-simulator for $\texttt{HyraxPC}.\mathsf{Eval}$ on input $(\mathsf{pp}, (C_a, r, C_v))$

Figure 5.15: Our $(r-1)$-ZK Simulator $\mathcal{S}$ for Lasso, where r is the number of rounds.

Lemma 5.6.3. *For all $\mu \in \mathbb{N}$, the sum-check protocol* `SumCheck` *in Fig. 5.13 is computational special sound, i.e., there exist a tree extractor $\mathcal{TE}_{\texttt{SumCheck}}$ and an EPT adversary $\mathcal{B}$ such that given an $\vec{n} := (1,2,2)^\mu$-tree of accepting transcripts (produced by an adversary $\mathcal{A}$) for the μ-rounds sum-check protocol, we have that:*

$$\mathbf{Adv}^{\mathsf{SS}}_{\texttt{SumCheck},\vec{n}}(\mathcal{TE}_{\mathsf{SC}}, \mathcal{A}) \leq \mu \left(\mathbf{Adv}^{\mathsf{DL}}_{\mathbb{G}}(\mathcal{B}) + \frac{1}{|\mathbb{F}|} \right)$$

Proof. We construct a tree extractor $\mathcal{TE}_{\texttt{SumCheck}}$ that does the following for each iteration $i \in [\mu]$. Given a $(1,1,2)$-tree of transcripts:

1. Run $\mathcal{TE}_{\texttt{DotProdPf}}$ on each $(1,1,2)$-subtree (corresponding to an instance $C_{p_i}, C_{y_i}, \vec{a}$) to extract $(p_i, \omega_{p_i}, y_i, \omega_{y_i})$, where y_i is supposedly equal to $e_{i-1} + w_i e_i$
2. Given two distinct challenges w_i, w_i', with extracted witnesses $(p_i, \omega_{p_i}, y_i, \omega_{y_i})$ and $(p_i', \omega_{p_i}', y_i', \omega_{y_i}')$ from the previous step, abort if $(p_i, \omega_{p_i}) \neq (p_i', \omega_{p_i}')$. Otherwise, solve for $e_{i-1}, e_i, \omega_{e_{i-1}}, \omega_{e_i}$ the system:

$$\begin{cases} y_i = e_{i-1} + w_i e_i \\ y_i' = e_{i-1} + w_i' e_i \end{cases} \qquad \begin{cases} \omega_{y_i} = \omega_{e_{i-1}} + w_i \omega_{e_i} \\ \omega_{y_i}' = \omega_{e_{i-1}} + w_i' \omega_{e_i} \end{cases}$$

The goal is to prove that $\mathcal{TE}_{\mathsf{SC}}$ either outputs polynomials $p_1(X), \ldots, p_\mu(X)$ that satisfy the information-theoretic sumcheck protocol, or we can build an adversary $\mathcal{B}$, as efficient as $\mathcal{TE}_{\texttt{SumCheck}}$ and $\mathcal{A}$ combined, against the discrete log problem in $\mathbb{G}$.

We have that $\langle p_i, a_i \rangle = y_i$ and $\langle p_i, a_i' \rangle = y_i'$ by the guarantees of $\mathcal{TE}_{\texttt{DotProdPf}}$. We derive that, if $\mathcal{TE}_{\texttt{SumCheck}}$ does not abort, it would extract values such that $p_i(0) + p_i(1) = e_{i-1}$ and $p_i(r_i) = e_i$, i.e., it extracts valid polynomials for the information-theoretic sumcheck protocol. In this case, we have that $C_{p_i} = \vec{g}^{p_i} \cdot h^{\omega_{p_i}} = \vec{g}^{p_i'} \cdot h^{\omega_{p_i}'}$, and by Lemma 2.3.1 we conclude that the probability to abort is bound by the probability to solve the discrete log problem in $\mathbb{G}$. By union bound on the number of rounds, we derive the claimed bound. □

Below we analyze the special soundness of `GrandProd` (cf. Fig. 5.14).

Lemma 5.6.4. *For all $d > 1$, the protocol* `GrandProd` *is computational special sound, i.e., there exist a tree extractor $\mathcal{TE}_{\texttt{GrandProd}}$ and EPT adversaries $\mathcal{B}, \mathcal{B}'$ such that given an $\vec{n}_{\texttt{GrandProd},d} :=$*

$((\vec{n}_0, 2, 2), \ldots, (\vec{n}_{d-2}, 2, 2))$-tree of accepting transcripts (produced by an adversary $\mathcal{A}$) for the grand product, we have:

$$\begin{aligned}\mathbf{Adv}^{\mathsf{SS}}_{\mathtt{GrandProd},\vec{n}}(\mathcal{TE}_{\mathtt{GrandProd}}, \mathcal{A}) \leq & \mathbf{Adv}^{\mathsf{SS}}_{\mathtt{HyraxPC.Open},((2^{d/2})_{:d/2},(4_{\pm})^{d/2},2)}(\mathcal{TE}_{\mathtt{HyraxPC}}, \mathcal{B}) \\ & + \sum_{i=1}^{d-2} 4^i \cdot \mathbf{Adv}^{\mathsf{SS}}_{\mathtt{SumCheck},(1,2,2)^i}(\mathcal{TE}_{\mathtt{SumCheck}}, \mathcal{B}')\end{aligned}$$

where $\vec{n}_0$ is the empty string and $\vec{n}_i := (4, 2, 2)^i$ for all $i > 0$.

Proof. We construct a tree extractor $\mathcal{TE}_{\mathtt{GrandProd}}$ that does the following.

1. For each iteration $i \in [d-1]$:
   (a) If $i > 1$, run $\mathcal{TE}_{\mathtt{SumCheck}}$ on each of the 4^i different $(1, 2, 2)^i$-subtrees, associated with the different random challenges $r_i^{(j)}$, to extract the polynomials sent during the sum-check
   (b) Run $\mathcal{TE}_{\mathtt{ProdPf}}$ on each 2-subtree to extract the values $(w_{1,i}^{(j)}, w_{2,i}^{(j)}, e_i^{(j)})$ and let f_i be the polynomial that interpolates all the pairs $(r_i^{(j)}, e_i^{(j)})$
2. Extract the polynomial $\widetilde{V}_d$ running $\mathcal{TE}_{\mathtt{HyraxPC}}$ on the subtree obtained by merging each $((4_{\pm})^{d/2}, 2)$-subtree corresponding to a different challenge point

At each iteration, the protocol `GrandProd` performs a sum-check to reduce the task of proving that a certain polynomial equals some claimed value over a hypercube of a given size, and a "reduction to a line" to batch two claims into one. Notice that the polynomial in the sum-check is only "virtually" represented and is never directly evaluated. We need to prove that at each iteration the prover performs a sum-check on a polynomial that is "consistent" wrt to the MLE of the input $\widetilde{V}_d$ that we extract using $\mathcal{TE}_{\mathtt{HyraxPC}}$.

We start by focusing on the last iteration of the protocol. Let $f_z(X) := \widetilde{eq}(z, X) \cdot \widetilde{V}_d(X, 0) \cdot \widetilde{V}(X, 1)$. We need to prove that the polynomial f_{d-1} extracted by $\mathcal{TE}_{\mathtt{GrandProd}}$ at the $(d-1)$-th iteration is equal to $f_z(X)$ for some z (corresponding to an ancestor of the current subtree).

First, by the guarantees of the information-theoretic sum-check and the special soundness of `ProdPf`, we have that $\sum_{p \in \{0,1\}^{d-2}} f_i(p) = v_{d-2}$, for a value v_{d-2} that has been committed at the previous iteration. Second, we observe that f_{d-1} is a $(d-1)$-variate polynomial of individual degree at most 3: this is because the polynomials sent during the sum-check are univariate polynomials of maximum degree 3, due to the number of `Pedersen` generators in `pp` and later used to run `ProdPf`. Moreover, by definition f_z is a $(d-1)$-variate polynomial of individual degree 3. Let Agree be the event that $f_{d-i}(r_{d-1}^{(j)}) = f_z(r_{d-1}^{(j)})$ for all j. Since there is a unique $(d-1)$-variate polynomial, of individual degree at most 3, that "densely" interpolates the pairs $(r_{d-1}^{(j)}, f_z(r_{d-1}^{(j)}))$ [Zip90], we conclude that, conditioned on Agree, $f_{d-1} \equiv f_z$. When Agree does not occur, we have that there is at least one challenge $r := r_{d-1}^{(j)}$ such that $f_{d-1}(r) \neq f_z(r)$. In particular, this implies that $w_{1,d-1}^{(j)} \neq \widetilde{V}_d(r, 0) \vee w_{2,d-1}^{(j)} \neq \widetilde{V}_d(r, 1)$. Let ℓ be the unique line interpolating $((r, 0), w_{1,d-1}^{(j)}), ((r, 1), w_{2,d-1}^{(j)})$; then, there exists at most one field element β such that $\ell(\beta) = \widetilde{V}_d(r, \beta)$. However, when Agree does not occur, we can find in the corresponding subtrees two distinct challenges $\beta_{d-1}^{(j)}, \beta_{d-1}^{\prime(j)}$ such that the above equation holds, from which we conclude that $\Pr[\mathsf{Agree}] = 1$.

A similar analysis can be run for all the layers of the circuit. We do not need to run $\mathcal{TE}_{\mathsf{SC}}$ when we reach the first iteration since the protocol does not invoke the `SumCheck` protocol. At the first layer, we only rely on the special soundness of `ProdPf` to extract the value v consistent with the output $\widetilde{V}_1(0)$. □

Lemma 5.6.5. *zkLasso satisfies $\vec{n}$-computational special soundness, where*

$$\vec{n} = (2^{\log m}, (2,2,2)^{\log m}, 2, (4_{\pm})^{(\log\log m)/2}, 2, 3, \mu+1, (\vec{n}_{\texttt{GrandProd},\mu})^{\alpha}, (4_{\pm})^{(\log\log m)/2}, 2)$$

Proof. We construct a tree extractor $\mathcal{TE}_{\texttt{Lasso}}$ that, given an $\vec{n}$-tree of accepting transcripts, does the following:

1. Run $\mathcal{TE}_{\texttt{SumCheck}}$ on the first sum-check sub-protocol on each $(1,2,2)^{\log m}$ subtree to extract the polynomials sent during the sum-check for $h(X)$
2. Run $\mathcal{TE}_{\texttt{GenPf}}$ on each corresponding 2-subtree to extract the values $v_1, \ldots, v_\alpha$ such that $\hat{f}(v_1, \ldots, v_\alpha) = e_x / \widetilde{eq}(r, r_z)$
3. Run $\mathcal{TE}_{\texttt{HyraxPC}}$ on the subtree obtained by merging each $((4_{\pm})^{\log m/2}, 2)$-subtree, corresponding to different challenge points, to extract α $\log m$-variate multilinear polynomials E_i such that $E_i(r_z) = v_i$ for all $i \in [\alpha]$
4. Run $\mathcal{TE}_{\texttt{GrandProd}}$ on each $\vec{n}_{\texttt{GrandProd},\mu}$-subtree to extract the multilinear polynomials dim_i, read_ts_i, write_ts_i, for all $i \in [\alpha]$, corresponding to the MLE of the last layer of the circuit
5. Output matrix M derived from the encoding of its non-zero entries in dim_i

We show that, conditioned on the event that none of the sub-extractor fails, the matrix M extracted by $\mathcal{TE}_{\texttt{Lasso}}$ is a valid witness. In particular, from the guarantees of $\mathcal{TE}_{\texttt{GrandProd}}$ and $\mathcal{TE}_{\texttt{SumCheck}}$ and the soundness of the corresponding protocols, we have that the prover unconditionally passes the verifier's checks for the sum-check and the memory checking argument (cf. Lemma 5.6.6) and, moreover, the rows of M are unit vectors. Also, from the guarantees of $\mathcal{TE}_{\texttt{SumCheck}}, \mathcal{TE}_{\texttt{HyraxPC}}$ and the soundness of the sum-check protocol we have that $M \cdot t = a$ because the check holds for more than $\log m$ random rows. □

We are ready to present our main theorem on zkLasso.

Theorem 5.6.1. *zkLasso is simulation-extractable.*

Proof. We prove that zkLasso is $(r-1)$-ZK and $(r-1)$-UR, where r is the number of rounds. By combining Theorem 5.3.1 and Lemma 5.6.5, we derive a direct proof of the theorem.

We leverage a simple $(r-1)$-ZK simulator $\mathcal{S}$ for zkLasso that, at high-level, executes all sub-protocols using a dummy witness and invokes the simulator for the final `HyraxPC.Eval`. For sake of completeness, we give the code of this simulator in Fig. 5.15.

First, by inspection, it is clear that the proofs produced are accepting: this is because the verifier accepts if both the Lasso proof (until the last round) is valid (let call this proof π_1), and if the final proof π_2 for `HyraxPC.Eval` is valid too. In fact, π_1 is a composition of honestly generated (sub)proofs, and by the completeness of Lasso, we derive that the verifier accepts all of them. The last proof π_2 is generated by invoking the ZK simulator for `HyraxPC.Eval`

(cf. Lemma 5.5.1), and in this case the validity follows from the NIZK guarantees. Second, we observe that $\mathcal{S}$ only makes a single RO reprogramming, in particular when invokes the ZK simulator for HyraxPC.Eval. Finally, we need to prove that the output of $\mathcal{S}$ is indistinguishable from that of a real transcript. Since the Lasso prover only employs hiding commitments as inputs to the inner sub-protocols and, additionally, all the sub-protocols used in Lasso are zero-knowledge, we conclude that the honestly generated proofs made by our simulator are identically distributed to the proofs in a real transcript. In the final (sub)protocol HyraxPC.Eval, indistinguishability is due to the guarantees of the ZK simulator.

The last sub-protocol of Lasso consists of an invocation of the $(\mu+1)$-rounds HyraxPC.Eval protocol that satisfies computational μ-UR (cf. Lemma 5.5.1). Hence, we conclude that zkLasso satisfies perfect $(r-1)$-UR. □

5.6.3 On Multi-Set Fingerprinting

Lemma 5.6.6. *The extracted multi sets in Step 9 of Fig. 5.11 are the same except with negligible probability.*

Proof. Let A and B be the two multisets and let $n := |A| = |B|$ (we can exclude that they have different cardinalities from the extraction procedure). Recall that each element in A or B is a tuple of three elements (x, v, t). In order to check their equality we check the equality of their fingerprints: $\mathcal{H}_{\tau,\gamma}(A) = \mathcal{H}_{\tau,\gamma}(B)$, where $\mathcal{H}_{\tau,\gamma}(A) = \prod_{(x,v,t)\in A}(h_\gamma(x,v,t) - \tau)$, and $h_\gamma(x,v,t) = x\cdot\gamma^2 + v\cdot\gamma + t$ denotes the Reed-Solomon fingerprinting.

Now assume that $A \neq B$ and let us bound the probability that the fingerprint test verifies. Below we denote by $\vec{\alpha}$ (resp. $\vec{\beta}$) the elements of the tuple $(h_\gamma(A_j))_{j\in[n]}$ (resp. B_j) where, in the indexing, we assume the elements of A (resp. B) are lexicographically ordered[9].

We observe that:

$$\begin{aligned}
&\Pr\left[\mathcal{H}_{\tau,\gamma}(A) = \mathcal{H}_{\tau,\gamma}(B)\right] \\
=&\Pr\left[\mathcal{H}_{\tau,\gamma}(A) = \mathcal{H}_{\tau,\gamma}(B) \wedge \vec{\alpha} \neq \vec{\beta}\right] + \Pr\left[\mathcal{H}_{\tau,\gamma}(A) = \mathcal{H}_{\tau,\gamma}(B) \wedge \vec{\alpha} = \vec{\beta}\right] \\
\leq&\Pr\left[\mathcal{H}_{\tau,\gamma}(A) = \mathcal{H}_{\tau,\gamma}(B) \mid \vec{\alpha} \neq \vec{\beta}\right] + \Pr\left[\vec{\alpha} = \vec{\beta}\right]
\end{aligned}$$

Intuitively the first summand in the last line refers to the event where the final product check passes *even though* the Reed-Solomon fingerprints somehow differ. The second summand in the last line is the probability that all the Reed-Solomon fingerprints are the same (despite A and B being distinct).

We observe:

$$\Pr\left[\mathcal{H}_{\tau,\gamma}(A) = \mathcal{H}_{\tau,\gamma}(B) \mid \vec{\alpha} \neq \vec{\beta}\right] = \Pr\left[\prod_j (\alpha_j - \tau) = \prod_j (\beta_j - \tau) \mid \vec{\alpha} \neq \vec{\beta}\right]$$

and in order to bound the last probability we can simply apply Schwartz-Zippel (the left- and right-hand side are two distinct polynomials of degree n evaluated in a random point τ) and conclude that it is lower or equal to $\frac{n}{|\mathbb{F}|}$.

[9] It could in fact be *any* canonical ordering. Having some ordering is going to simplify some observations in our proof.

We now bound the other summand. First, we observe that, from our assumption $A \neq B$ there must exist some index j^* such that $A_{j^*} \neq B_{j^*}$ (recall we are assuming a lexicographic ordering of the multisets). Therefore:

$$\Pr\left[\vec{\alpha} = \vec{\beta}\right] = \Pr\left[\bigwedge_j \alpha_j = \beta_j\right] \leq \Pr\left[\alpha_{j^*} = \beta_{j^*}\right]$$

By the definition of h_γ we can then apply Schwartz-Zippel again and conclude that the last event can occur with probability at most $\frac{2}{|\mathbb{F}|}$.

We showed that if $A \neq B$ then the probability that the test passes is at most $\frac{n}{|\mathbb{F}|} + \frac{2}{|\mathbb{F}|}$. Since this quantity is negligible this concludes the proof. □

5.7 Modular Composition of Non-malleable Arguments

We describe two variations of two compilers for modular compositions of non-interactive arguments of knowledge. The first two compilers handle conjunction of relations with shared witness; the other two handle functional compositions.

5.7.1 General Results on Conjunction and Functional Composition

In both cases, the compilers start from commit-and-prove arguments that are simulation-extractable. However, for two of the compilers, we require the slightly more general notion of signature-of-knowledge.

Definition 5.7.1. *We say that a non-interactive argument* Π *is a signature-of-knowledge for a relation* $\mathcal{R}$*, if* Π *is a complete, simulation extractable and zero-knowledge non-interactive argument for the (augmented) relation* $\mathcal{R}'$ *such that:*

$$\forall \mathtt{msg} \in \{0,1\}^\lambda : \mathcal{R}(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \iff \mathcal{R}'(\mathsf{pp}, (\mathtt{msg}, \mathbb{x}), \mathbb{w}),$$

where msg *is referred to as the* signed message*.*

(Generalized) Conjunction of arguments. We consider two compilers for conjunction of relations with common witnesses with different trade-offs. Additionally, we generalize the notion of conjunction with common witness by assuming a (possible) processing through a function M to such a common witness. Specifically, given relation $\mathcal{R}_A$ and $\mathcal{R}_B$ we define $\mathcal{R}^M_{A\wedge B}$ the relation such that $\mathcal{R}^M_{A\wedge B}(\mathsf{pp}, \mathbb{x}_A, \mathbb{x}_B, \mathbb{w}) \iff \mathcal{R}_A(\mathsf{pp}, \mathbb{x}_A, \mathbb{w}) \wedge \mathcal{R}_B(\mathsf{pp}, \mathbb{x}_B, M(\mathbb{w}))$.

Definition 5.7.2. *Let* M *be a polynomial time function, we say that a commitment scheme* CS *is* M*-malleable if there exist efficiently computable functions* M_c, M_ρ *such that, for any commitment* c *to* $\mathbb{w}$ *with opening* ρ *we have that* $M_c(\mathsf{c})$ *is a valid commit to* $M(\mathbb{w})$ *with opening* $M_\rho(\rho)$*. Namely,* $\forall \mathsf{pp}, \mathsf{c}, \mathbb{w}, \rho : \mathsf{VerCom}(\mathsf{pp}, \mathsf{c}, \mathbb{w}, \rho) \Rightarrow \mathsf{VerCom}(\mathsf{pp}, M_c(\mathsf{c}), M(\mathbb{w}), M_\rho(\rho))$.

We define a compiler from simulation-extractable arguments (resp. signature-of-knowledge) $\Pi_\wedge$ (resp. $\bar{\Pi}_\wedge$) for $\mathcal{R}^M_{A\wedge B}$ in Fig. 5.16.

Functional composition of arguments. For any polynomial-time function f let the relation

Domain separation: H_1 and H_2 are two random oracles.

// Conjunction proofs $\Pi_\wedge$ and $\bar{\Pi}_\wedge$ (including boxed instructions)

Prover: $\mathcal{P}^{\mathsf{H}_1,\mathsf{H}_2}(\mathsf{pp}, (\mathbb{x}_A, \mathbb{x}_B), \mathbb{w})$ does:

1. commit $\mathsf{c}, \rho \leftarrow\$ \mathsf{CS.Com}(\mathsf{ck}, \mathbb{w})$,
2. prove $\pi_A \leftarrow\$ \Pi_A.\mathcal{P}^{\mathsf{H}_1}(\mathsf{pp}, (\mathsf{c}, \mathbb{x}_A), (\mathbb{w}, \rho))$,
3. prove $\pi_B \leftarrow\$ \Pi_B.\mathcal{P}^{\mathsf{H}_2}(\mathsf{pp}, (\boxed{\mathtt{msg}}, M_c(\mathsf{c}), \mathbb{x}_B), (M(\mathbb{w}), M_\rho(\rho)))$
   where $\mathtt{msg} := \mathbb{x}_A \| \pi_A$

Verifier: $\mathsf{Verify}^{\mathsf{H}_1,\mathsf{H}_2}(\mathsf{pp}, (\mathbb{x}_A, \mathbb{x}_B), \pi)$ parses $\pi = (\mathsf{c}, \pi_A, \pi_B)$, return 1 if and only if $\mathsf{Verify}^{\mathsf{H}_1}(\mathsf{pp}, (\mathsf{c}, \mathbb{x}_A), \pi_A) = 1$ and $\mathsf{Verify}^{\mathsf{H}_2}(\mathsf{pp}, (\boxed{\mathtt{msg},} M_c(\mathsf{c}), \mathbb{x}_B), \pi_B) = 1$ // Composition proofs $\Pi_{g\circ f}$ and $\bar{\Pi}_{g\circ f}$ (including boxed instructions)

Prover: $\mathcal{P}^{\mathsf{H}_1,\mathsf{H}_2}(\mathsf{c}_i, \mathsf{c}_o, \mathbb{x}_i, \mathbb{x}_o, \mathbb{w}_i, \mathbb{w}_o, \rho_i, \rho_o)$ does

1. parse $\mathbb{x}_i = x_{f,i} \| x_{g,i}$ and $\mathbb{x}_o = x_{f,o} \| x_{g,o}$ and $x_f = (x_{f,i}, x_{f,o})$,
2. let $(x_{f,o}, w_{g,i}) \leftarrow f(x_{f,i}, w_{f,i})$ and $\mathsf{c}', \rho' \leftarrow \mathsf{CS.Com}(\mathsf{ck}, w_{g,i})$
3. prove $\pi_F \leftarrow \Pi_f.\mathcal{P}^{\mathsf{H}_1}(\mathsf{c}_i, \mathsf{c}', (x_{f,i}, x_{F_o}), (\mathbb{w}_i, w_{g,i}), \rho_i, \rho')$,
4. prove $\pi_G \leftarrow \Pi_g.\mathcal{P}^{\mathsf{H}_2}(\boxed{\mathtt{msg}}, \mathsf{c}', \mathsf{c}_o, (x_{g,i}, x_{g,o}), (w_{g,i}, w_o), \rho', \rho_o)$
   where $\mathtt{msg} := \mathsf{c}_i \| x_F \| \pi_1$

Verifier: $\mathsf{Verify}((\mathsf{c}_i, \mathsf{c}_o, \mathbb{x}_i, \mathbb{x}_o), \pi)$ parses $\pi = (\mathsf{c}', \pi_1, \pi_2)$, return 1 if and only if $\Pi_f.\mathsf{Verify}^{\mathsf{H}_1}((\mathsf{c}_i, \mathsf{c}', x_{f,i}, x_{f,o}), \pi_1) = 1$ and $\Pi_g.\mathsf{Verify}^{\mathsf{H}_2}((\mathtt{msg}, \mathsf{c}', \mathsf{c}_o, x_{g,i}, x_{g,o}), \pi_2) = 1$.

Figure 5.16: Compiler to proofs for conjunction (top) and function composition (bottom). For $X \in \{A, B, f, g\}$ Π_X is assumed to be a commit-and-prove non-interactive argument over commitment scheme CS (assumed to be M-malleable for the compiler for conjunction). For $\bar{\Pi}_\wedge$ (resp. $\bar{\Pi}_{g\circ f}$) we additionally assume that Π_A (resp. Π_g) is a signature of knowledge.

$\mathcal{R}_f$ be such that $\mathcal{R}_f(\mathsf{pp}, (\mathbb{x}_i, \mathbb{x}_o), (\mathbb{w}_i, \mathbb{w}_o)) \iff f(\mathbb{x}_i, \mathbb{w}_i) = (\mathbb{x}_o, \mathbb{w}_o)$. We define $g \circ f$ to be the functional composition of g and f, namely, the function that on input $((x_{f,i}, x_{g,i}), w_{f,i})$ computes $(x_{f,o}, w_{g,i}) \leftarrow f(x_{f,i}, w_{f,i})$, computes $(x_{g,o}, \mathbb{w}_o) \leftarrow g(x_{g,i}, w_{g,i})$ and outputs $((x_{f,o}, x_{g,o}), \mathbb{w}_o)$. See Fig. 5.17 for a graphical representation of functional composition.

We define a compiler to functional composition from simulation-extractable arguments $\Pi_{g\circ f}$ and from a signature of knowledge $\bar{\Pi}_{g\circ f}$ for $\mathcal{R}_{g\circ f}$ in Fig. 5.16.

Additional Definitions and Theorem on Compilers Security. We are almost ready to state the theorem. We first need two additional definitions.

Definition 5.7.3. *We say that a relation $\mathcal{R}$ is efficiently witness computable if there exists an EPT algorithm $\mathcal{M}$ such that for any pp and $\mathbb{x}$ we have either $\mathcal{R}(\mathsf{pp}, \mathbb{x}, \mathcal{M}(\mathsf{pp}, \mathbb{x})) = 1$ or $(\mathsf{pp}, \mathbb{x}) \notin \mathcal{L}_\mathcal{R}$. We say that a relation $\mathcal{R}$ is always satisfiable if, for any pp, the language $\mathcal{L}_{\mathcal{R},\mathsf{pp}} = \{0,1\}^*$, where the latter is the language associated to the relation for given parameters pp.*

The definition above indicates that the relation $\mathcal{R}$ can be decided by an expected polynomial-time algorithm. At first glance, one might consider an argument of knowledge for a relation

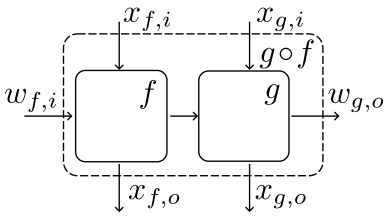


Figure 5.17: Graph for the functional composition $g \circ f$.

in P[10] to be somewhat trivial. However, the scenario becomes more compelling in the context of commit-and-prove relations. In this case, while $\mathcal{R}$ is decidable, the corresponding commit-and-prove relation $\hat{\mathcal{R}}$ is not, unless, we allow the prover to sample the commitment to the witness.

Nicely, when the relation $\mathcal{R}_A$ (resp. $\mathcal{R}_f$) efficiently witness computable we can weaken the zero-knowledge property of Π_A (resp. Π_f) in the compilers to witness indistinguishability (WI)[11]. Furthermore, for WI to hold, it is not necessary to reprogram the random oracle.

Definition 5.7.4. *A non-interactive argument for $\mathcal{R}$ is statistically witness-indistinguishable (WI) if for any pp and any $\mathbb{x}, \mathbb{w}_1, \mathbb{w}_2$ such that $(\mathsf{pp}, \mathbb{x}, \mathbb{w}_i) \in \mathcal{R}$ the distributions $\mathcal{P}^{\mathsf{H}}(\mathsf{pp}, \mathbb{x}, \mathbb{w}_i)$ for $i \in \{1, 2\}$ are statically close.*

Theorem 5.7.1. *Assuming that the commitment scheme CS is hiding and binding, the following statements hold true:*

1. *For any PT M, if CS is M-malleable, and Π_A and Π_B are trapdoorless zero-knowledge and simulation extractable then $\Pi_\wedge$ for the relation $\mathcal{R}^M_{A \wedge B}$ is simulation-extractable.*

2. *If Π_f and Π_g are trapdoorless zero-knowledge and simulation extractable then $\Pi_{g \circ f}$ is simulation-extractable.*

3. *For any PT M, CS is M-malleable, and if Π_A is knowledge sound and statistically witness indistinguishable, $\mathcal{R}_A$ is always satisfiable and efficiently witness computable and Π_B is trapdoorless zero-knowledge and a signature-of-knowledge then Π' is simulation-extractable.*

4. *If Π_f is knowledge sound and statistically witness indistinguishable, $\mathcal{R}_f$ is always satisfiable and efficiently witness computable or the public output of f is the empty string, namely for any $x_{f,i}, \mathbb{w}_i$ we have $|x_{f,o}| = 0$ where $x_{f,o}, w_{f,o} = f(x_{f,i}, \mathbb{w}_i)$, and Π_g is trapdoorless zero-knowledge and a signature-of-knowledge then $\bar{\Pi}_{g \circ f}$ is simulation-extractable.*

Before proving the theorem we remark that the notion of *trapdoorless* zero-knowledge is key for the four statements to hold. This is evident, for example, in the proof of the fourth statement, where we can invoke the knowledge soundness of Π_f in the presence of simulated proofs for Π_g. We can do this because to simulate proofs we only need to reprogram the random oracle H_2 which does not interfere with Π_f. On the other hand, if we needed a trapdoor for the simulations then we would need to make sure that the knowledge sound of Π_f held in the presence of such a trapdoor (for example, by sampling independent reference strings for the two schemes, which is unnatural and cumbersome in many practical scenarios).

[10] More precisely, the class $AvgP$.

[11] Zero-knowledge implies witness indistinguishability, see Feige and Shamir [FS90].

Proof of Theorem 5.7.1. We proceed statement by statement.

First statement. Completeness of Π_A and Π_B, together with the M-malleability of the commitment scheme, imply the completeness of $\Pi_\wedge$.

As for simulation extractability, let $\mathcal{P}^*$ be an adversary for the simulation extractability of $\Pi_\wedge$. Given the set $\mathcal{Q}_{\mathsf{sim}}$ of queries and answers to the simulation oracle, we can derive the set $\mathcal{Q}_{\mathsf{sim},i}$ of queries and answers to Π_i for $i \in \{A, B\}$. Specifically, if $(\mathbb{x}, \pi) \in \mathcal{Q}_{\mathsf{sim}}$ and $\mathbb{x} = (\mathbb{x}_A, \mathbb{x}_B)$ and $\pi = (\mathsf{c}, \pi_A, \pi_B)$ then $((\mathsf{c}, \mathbb{x}_i), \pi_i) \in \mathcal{Q}_{\mathsf{sim},i}$ for $i \in \{A, B\}$. As a shortcut, given a tuple $(\mathbb{x}, \pi)$ for $\Pi_\wedge$, we can define $(\mathbb{x}, \pi)_i$ the derived tuple of instance and proof for Π_i.

Let $(\tilde{\mathbb{x}}, \tilde{\pi})$ be the forgery of the adversary, where $\tilde{\mathbb{x}} := (\tilde{\mathbb{x}}_A, \tilde{\mathbb{x}}_B)$ and $\tilde{\pi} := (\tilde{\mathsf{c}}, \tilde{\pi}_1, \tilde{\pi}_B)$, and consider the event bad:

$$(\tilde{\mathbb{x}}, \tilde{\pi}) \notin \mathcal{Q}_{\mathsf{sim}} \wedge \left(\bigwedge_{i \in \{A,B\}} (\tilde{\mathbb{x}}, \tilde{\pi})_i \in \mathcal{Q}_{\mathsf{sim},i} \right) \tag{5.5}$$

It is easy to check that the $\Pr[\mathsf{bad}] = 0$. In fact, if $(\tilde{\mathbb{x}}, \tilde{\pi})_1 \in \mathcal{Q}_{\mathsf{sim},1}$ then either $\tilde{\mathbb{x}}_B$ or $\tilde{\pi}_B$ are *fresh*, namely either $\forall \pi_2' : (\tilde{\mathsf{c}}, \tilde{\mathbb{x}}_B, \pi_2') \notin \mathcal{Q}_{\mathsf{sim},2}$ or $\forall \mathbb{x}_2' : (\tilde{\mathsf{c}}, \mathbb{x}_2', \tilde{\pi}_B) \notin \mathcal{Q}_{\mathsf{sim},2}$, as otherwise $(\tilde{\mathbb{x}}, \tilde{\pi}) \in \mathcal{Q}_{\mathsf{sim}}$. The other alternative is $(\tilde{\mathbb{x}}, \tilde{\pi})_2 \in \mathcal{Q}_{\mathsf{sim},2}$, which is handled similarly.

First we show that $\Pi_\wedge$ is indeed zero-knowledge. Let $\mathcal{S}_i$ be the zero-knowledge simulator for Π_i, and consider the zero-knowledge simulator $\mathcal{S}_\wedge$ that runs $\mathcal{S}_A, \mathcal{S}_B$ in parallel, in particular the simulator provides three interfaces to the adversary, the simulation oracle query on the appropriate instances the simulators $\mathcal{S}_A$ and $\mathcal{S}_B$ (following the specification as in the prover), while the other two oracles are the simulator for the random oracles, in particular, $\mathcal{S}_\wedge$ queries $\mathcal{S}_A$ for the queries directed to H_1 and $\mathcal{S}_B$ for the queries directed to H_2. Because of the domain separation, the simulators can handle the RO-queries independently. More in detail, the simulation oracles are handled as follows:

- Parse the instance as $\mathbb{x}$ as $(\mathbb{x}_A, \mathbb{x}_B)$.
- Sample a commitment to a dummy value, namely $\mathsf{c}, \rho \leftarrow \mathsf{CS.Com}(\mathsf{ck}, \bar{0})$.
- Run the simulator $\mathcal{S}_A$ on $(\mathsf{c}, \mathbb{x}_A)$ and $\mathcal{S}_B$ on $(M_c(\mathsf{c}), \mathbb{x}_B)$.

It is rather straight-forward to show that if Π_A and Π_B are zero-knowledge and the commitment scheme is hiding, then $\Pi_\wedge$ is zero-knowledge.

Let $\mathcal{P}_i^*$ be the adversary for the simulation extractability of Π_i that internally runs $\mathcal{P}^*$ and the simulator $\mathcal{S}_{\bar{i}}$ where $i \in \{A, B\}$ and $\bar{i}$ is set to B if $i = A$ and to B otherwise. Specifically, the adversary does:

- Upon simulation query $\mathbb{x}$ for $\Pi_\wedge$, similarly to the simulator $\mathcal{S}$, it samples a commitment $\mathsf{c}, \rho \leftarrow \mathsf{CS.Com}(\mathsf{ck}, \bar{0})$, runs $\mathcal{S}_{\bar{i}}$ on the derived instance $(\mathsf{c}, \mathbb{x}_{\bar{i}})$ and queries the simulation oracle for the instance $(\mathsf{c}, \mathbb{x}_i)$.
- Forward the query to the random oracle appropriately: either internally handled by $\mathcal{S}_{\bar{i}}$, or externally forwarded the queries to $\mathcal{P}_i^*$'s random oracle.
- Upon forgery $(\tilde{\mathbb{x}}, \tilde{\pi})$ output the forgery $(\tilde{\mathbb{x}}, \tilde{\pi})_i$.

Since the event bad defined in Eq. (5.5) never happens, from a valid forgery for $\Pi_\wedge$ we can derive either valid forgery for Π_A or for Π_B, thus:

$$\Pr\left[\mathsf{SE}^{\mathcal{S},\mathcal{P}^*}_{0,\Pi_\wedge}(\lambda)\right] \leq \Pr\left[\mathsf{SE}^{\mathcal{S}_A,\mathcal{P}^*_A}_{0,\Pi_A}(\lambda)\right] + \Pr\left[\mathsf{SE}^{\mathcal{S}_B,\mathcal{P}^*_B}_{0,\Pi_B}(\lambda)\right] \tag{5.6}$$

We define the knowledge extractor $\mathcal{E}$ for $\Pi_\wedge$:

- For $i \in \{A, B\}$, run $\mathcal{E}_i$ interacting with $\mathcal{P}^*_i$ and let $\hat{\mathbb{w}}_i := (\mathbb{w}_i, \rho_i)$ be the output of $\mathcal{E}_i$.
- If $(M(\mathbb{w}_A), M_o(\rho_A)) \neq (\mathbb{w}_B, \rho_B)$ abort, otherwise output $\mathbb{w}_A$.

Notice, the description above is incomplete because we did not describe how $\mathcal{E}$ provides the interaction between $\mathcal{E}_i$ and $\mathcal{P}^*_i$. More in detail, the extractor can provide a virtual interface to $\mathcal{P}^*_i$ given oracle access to $\mathcal{P}^*$ using the same strategy we define the adversary $\mathcal{P}^*_i$ using only oracle access to $\mathcal{P}^*$. Moreover, the two (internal) extractors are run with independent randomness. We can show that:

$$\begin{aligned}\Pr\left[\mathsf{SE}^{\mathcal{E},\mathcal{S},\mathcal{P}^*}_{1,\Pi_\wedge}(\lambda)\right] &= \Pr\left[\mathsf{SE}^{\mathcal{S},\mathcal{P}^*}_{0,\Pi_\wedge}(\lambda) \wedge \neg\left(\wedge_i(\tilde{\mathsf{c}}, \tilde{\mathbb{x}}_i, \hat{\mathbb{w}}_i) \in \hat{\mathcal{R}}_i \wedge \neg\mathtt{Abort}\right)\right] \\ &\leq \sum_i \Pr\left[\mathsf{SE}^{\mathcal{E}_i,\mathcal{S}_i,\mathcal{P}^*_i}_{1,\Pi_i}(\lambda)\right] + \Pr[\mathsf{Abort}]\end{aligned} \tag{5.7}$$

The running time of the extractor is the sum of the running times of $\mathcal{E}_A$ and $\mathcal{E}_B$, and when Abort happens, we can break the binding property of the commitment scheme. Putting Eqs. (5.6) and (5.7) together we have:

$$\mathbf{Adv}^{\mathsf{SIM-EXT}}_{\Pi_\wedge,\mathcal{R}}(\mathcal{S},\mathcal{E},\mathcal{P}^*) \leq \sum_{i\in\{A,B\}} \mathbf{Adv}^{\mathsf{SIM-EXT}}_{\Pi_i,\mathcal{R}_i}(\mathcal{S}_i,\mathcal{E}_i,\mathcal{P}^*_i) + \mathsf{negl}(\lambda)$$

Second Statement. The proof for this statement is almost the same as the previous proof. The main difference is the definition of the extractor which aborts in case it finds two different openings for the commitment c'. We give more details on the extractor in the proof of the forth statement.

Third Statement. Similarly to the proof of the first statement, we consider $\mathcal{P}^*$ be an adversary for the simulation extractability of $\bar{\Pi}_\wedge$. Given the set $\mathcal{Q}_{\mathsf{sim}}$ of queries and answers to the simulation oracle, we can derive the set $\mathcal{Q}_{\mathsf{sim},i}$ of queries and answers to Π_i for $i \in \{A, B\}$. The only difference is that from a tuple $(\mathbb{x}, \pi)$ for $\bar{\Pi}_\wedge$ we derive the tuple $(\mathbb{x}, \pi)_B = (\mathtt{msg}, \mathsf{c}, \mathbb{x}_B, \pi_B)$ where $\mathtt{msg} = \mathbb{x}_A \| \pi_A$ for Π_B. Thanks to this difference, if the forgery $(\tilde{\mathbb{x}}, \tilde{\pi}) \notin \mathcal{Q}_{\mathsf{sim}}$ then the derived forgery $(\tilde{\mathbb{x}}, \tilde{\pi})_2 \notin \mathcal{Q}_{\mathsf{sim},2}$.

First we show that $\bar{\Pi}_\wedge$ is indeed zero-knowledge. Let $\mathcal{S}_B$ be the zero-knowledge simulator for Π_B, and consider the zero-knowledge simulator $\mathcal{S}_\wedge$ that:

- Parse the instance as $\mathbb{x} = (\mathbb{x}_A, \mathbb{x}_B)$.
- Let $\mathcal{M}$ be the algorithm satisfying the efficient witness computability of $\mathcal{R}_A$, compute $\mathbb{w}_A \leftarrow \mathcal{M}(\mathbb{x}_A)$.
- Sample ρ_A and computes honest proof π_A for $(\mathsf{c}, \mathbb{x}_A, \mathbb{w}, \rho_A) \in \hat{\mathcal{R}}_A$ where $\mathsf{c}, \rho_A \leftarrow \mathsf{CS.Com}(\mathsf{ck}, \mathbb{w}_A)$.
- Run the simulator $\mathcal{S}_B$ on $(\mathtt{msg}, \mathsf{c}, \mathbb{x}_B)$.

We show that if Π_A is statistical witness-indistinguishable, Π_B is zero-knowledge, and the commitment scheme is hiding, then $\Pi_\wedge$ is zero-knowledge. We start from the real-world distribution of honestly generated proofs and move to the ideal distribution of simulated proofs through a hybrid argument.

- The first hybrid $\mathbf{H}_1$ is the same as the real-world distribution but the proof π_B is computed using the simulator $\mathcal{S}_B$ on message $(\mathtt{msg}, \mathsf{c}, \mathbb{x}_B)$. It is easy to show that the real world and the hybrid $\mathbf{H}_1$ are statistically close thanks to the (statistical) zero-knowledge property of Π_B.

- In the second hybrid $\mathbf{H}_2$, the prover, on input $(\mathbb{x}, \mathbb{w})$, additionally computes $\mathbb{w}' \leftarrow \mathcal{M}(\mathbb{x}_A)$ and breaks the binding of the commitment c finding ρ' such that $\mathsf{VerCom}(\mathsf{ck}, \mathsf{c}, \mathbb{w}', \rho') = 1$. It aborts if it cannot find such an opening ρ'. The difference between the two hybrids is the event that $\mathbf{H}_2$ might abort. We can show that, since the commitment scheme is statistically hiding, the event happens with negligible probability. Briefly, the reduction fixes messages $\mathbb{w}$ and $\mathbb{w}' \leftarrow \mathcal{M}(\mathbb{x}_A)$ and, given a challenge commitment c, it outputs 1 if it can brute-force the commitment on a valid opening w.r.t. $\mathbb{w}'$. Notice, if the challenge commitment is a commitment to $\mathbb{w}$, the reduction outputs 0 with the same probability of the aborting event, while if c is a commitment to $\mathbb{w}'$ there always exists a valid opening so the reduction eventually outputs 1.

- The hybrid $\mathbf{H}_3$ is the same as $\mathbf{H}_2$ but the proof π_A is computed using witness $(\mathbb{w}', \rho')$. The two hybrid are statistically close thanks to the statistical witness indistinguishability of Π_A.

- The last hybrid $\mathbf{H}_4$ is the same as $\mathbf{H}_3$, but the commitment is computed directly as a commitment to $\mathbb{w}'$. Again, we can reduce to the hiding of the commitment scheme. Also notice, this last hybrid is equivalent to the simulated world.

We are ready to prove simulation extractability. Let $\mathcal{P}_A^*$ be an adversary for the knowledge extractability of Π_A that internally runs $\mathcal{P}^*$ and $\mathcal{S}_2$. Notice that, even if $\mathcal{R}_A$ is efficiently witness computable, the relation $\hat{\mathcal{R}}_A$ (proved by Π_A) is not polynomially decidable, thus the notion of knowledge extractability is still meaningful.

Specifically the adversary $\mathcal{P}_A^*$ does:

- Upon simulation query $\mathbb{x}$ for $\bar{\Pi}_\wedge$, similarly to the simulator $\mathcal{S}$ described above, it computes $\mathbb{w}_A \leftarrow \mathcal{M}(\mathbb{x}_A)$ and samples commitment $\mathsf{c}, \rho_A \leftarrow \mathsf{CS.Com}(\mathsf{ck}, \mathbb{w}_A)$, it runs $\mathcal{S}_2$ on the derived instance $(\mathtt{msg}, \mathsf{c}, \mathbb{x}_2)$.

- Internally forward the query to H_2 to $\mathcal{S}_B$, and (externally) forward the queries to H_1.

- Upon forgery $(\tilde{\mathbb{x}}, \tilde{\pi})$ output the forgery $(\tilde{\mathbb{x}}, \tilde{\pi})_A$.

Almost identically, let $\mathcal{P}_B^*$ be an adversary for the simulation extractability of Π_B that internally runs $\mathcal{P}^*$. Specifically the adversary $\mathcal{P}_B^*$ does:

- Upon simulation query $\mathbb{x}$ for $\bar{\Pi}_\wedge$, similarly to the simulator $\mathcal{S}$ described above, it computes $\mathbb{w}_A \leftarrow \mathcal{M}(\mathbb{x}_A)$ and samples commitment $\mathsf{c}, \rho_A \leftarrow \mathsf{CS.Com}(\mathsf{ck}, \mathbb{w}_A)$, and it queries the simulation oracle on the derived instance $(\mathtt{msg}, \mathsf{c}, \mathbb{x}_B)$.

- Forward the query to H_i for $i \in \{A, B\}$ to the appropriate oracles.
- Upon forgery $(\tilde{\mathbb{x}}, \tilde{\pi})$ output the forgery $(\tilde{\mathbb{x}}, \tilde{\pi})_B$.

Identically to the proof of the first statement, we define the knowledge extractor $\mathcal{E}$ for $\bar{\Pi}_\wedge$ to:

- For $i \in \{A, B\}$, run $\mathcal{E}_i$ interacting with $\mathcal{P}_i^*$ and let $\hat{\mathbb{w}}_i = (\mathbb{w}_i, \rho_i)$ be the output of $\mathcal{E}_i$.
- If $M(\mathbb{w}_A), M_o(\rho_A) \neq \mathbb{w}_B, \rho_B$ abort, otherwise output $\mathbb{w}_A$.

As in the proof of the first statement, the extractor can provide a virtual interface to $\mathcal{P}_i^*$ given oracle access to $\mathcal{P}^*$ using the same strategy we define the adversary $\mathcal{P}_i^*$ using only oracle access to $\mathcal{P}^*$. The only difference is that $\mathcal{P}_1^*$ is an adversary for the knowledge extractability (it does not need simulation queries). Moreover, the two (internal) extractors are run with independent randomness. It is rather straight-forward to show that:

$$\begin{aligned}\Pr\left[\mathsf{SE}_{1,\bar{\Pi}_\wedge}^{\mathcal{E},\mathcal{S},\mathcal{P}^*}(\lambda)\right] &= \Pr\left[\mathsf{SE}_{0,\bar{\Pi}_\wedge}^{\mathcal{S},\mathcal{P}^*}(\lambda) \wedge \neg\left(\wedge_i(\tilde{\mathsf{c}}, \tilde{\mathbb{x}}_i, \hat{\mathbb{w}}_i) \in \hat{\mathcal{R}}_i \wedge \neg\mathsf{Abort}\right)\right] \leq \\ &\leq \Pr\left[\mathsf{SE}_{1,\Pi_B}^{\mathcal{E}_B,\mathcal{S}_B,\mathcal{P}_B^*}(\lambda)\right] + \Pr\left[\mathsf{KS}_{A,\Pi_A}^{\mathcal{E}_A,\mathcal{P}_A^*}(\lambda)\right] + \Pr[\mathsf{Abort}] \qquad (5.8)\end{aligned}$$

The running time of the extractor is the sum of the running times of $\mathcal{E}_A$ and $\mathcal{E}_B$, and when Abort happens, we can break the binding property of the commitment scheme. Putting things together we have:

$$\mathbf{Adv}_{\bar{\Pi}_\wedge,\mathcal{R}}^{\mathsf{SIM-EXT}}(\mathcal{S}, \mathcal{E}, \mathcal{P}^*) \leq \mathbf{Adv}_{\Pi_A,\mathcal{R}_A}^{\mathsf{KS}}(\mathcal{E}_A, \mathcal{P}_A^*) + \mathbf{Adv}_{\Pi_B,\mathcal{R}_B}^{\mathsf{SIM-EXT}}(\mathcal{S}_B, \mathcal{E}_B, \mathcal{P}_B^*) + \mathsf{negl}(\lambda). :$$

Fourth Statement. Given the set $\mathcal{Q}_{\mathsf{sim}}$ of queries and answers to the simulation oracle, we can derive the sets $\mathcal{Q}_{\mathsf{sim},f}$ and $\mathcal{Q}_{\mathsf{sim},g}$ of queries and answers to Π_f and Π_g respectively. Specifically, if $\mathbb{x}, \pi \in \mathcal{Q}_{\mathsf{sim}}$ and $\mathbb{x} = (\mathsf{c}_i, \mathsf{c}_o, \mathbb{x}_i, \mathbb{x}_o)$ and $\pi = (\mathsf{c}', \pi_f, \pi_g)$ then $(\mathsf{c}_i, \mathsf{c}', x_{f,i}, x_{f,o}), \pi_f \in \mathcal{Q}_{\mathsf{sim},f}$ and $(\mathtt{msg}, \mathsf{c}', \mathsf{c}_o, x_{g,i}, x_{g,o}), \pi_g \in \mathcal{Q}_{\mathsf{sim},g}$ where $\mathtt{msg} = \mathsf{c}' \| x_f \| \pi_f$. As a shortcut, given a tuple $\mathbb{x}, \pi$ for $\Pi_{g \circ f}$, we can define $(\mathbb{x}, \pi)_X$ the derived tuple of instance and proof for Π_X for $X \in \{f, g\}$.

Similarly to the simulator for zero-knowledge in the proof of the third statement. We can define a simulator for $\Pi_{g \circ f}$ that makes use of the efficient witness computability of $\mathcal{R}_f$. Additionally, we give a second simulator for the special case where $x_{f,o}$ is the empty string for any assignments of the public and private inputs $x_{f,i}, w_{f,i}$.

Let $\mathcal{S}_g$ the zero-knowledge simulator for Π_g, and consider the zero-knowledge simulator $\mathcal{S}$ (resp. the zero-knowledge simulator $\mathcal{S}'$ that executes Item 2b instead of Item 2a) that:

1. Parse the instance as $\mathbb{x} = (\mathbb{x}_f, \mathbb{x}_g)$.
2. Execute one of the two steps:
   (a) Let $\mathcal{M}$ be the algorithm satisfying the efficient witness computability of $\mathcal{R}_f$, compute $\mathbb{w}_f = (w_{f,i}, w_{f,o}) \leftarrow \mathcal{M}(\mathbb{x}_f)$.
   (b) Set $w_{f,i} := \bar{0}$ compute $w_{f,o} \leftarrow f(x_{f,i}, w_{f,o})$ and let $\mathbb{w}_f := (w_{f,i}, w_{f,o})$.
3. Compute honest proof π_F for $(\mathsf{c}_i, \mathsf{c}', \mathbb{x}_f, \mathbb{w}_f) \in \hat{\mathcal{R}}_f$ where $\mathsf{c}', \rho' \leftarrow \mathsf{CS.Com}(\mathsf{ck}, w_{f,o})$ and $\mathsf{c}_i, \rho_i \leftarrow \mathsf{CS.Com}(pp, w_{f,i})$.
4. Sample dummy commitment $\mathsf{c}_o, \rho_o \leftarrow \mathsf{CS.Com}(\mathsf{ck}, \bar{0})$, run the simulator $\mathcal{S}_g$ on $(\mathtt{msg}, \mathsf{c}', \mathsf{c}_o, \mathbb{x}_g)$.

The proofs of zero-knowledge w.r.t. the two simulators follow similarly to the proof of zero-knowledge in the third statement. In particular, in both cases, we first switch to simulated proofs for Π_g and then use the hybrid argument that use a combination of the hiding property and the witness indistinguishability property. Also to prove simulation extractability we proceed similarly to the proof of the third statement. We omit the details as the proof is almost identical. □

5.7.2 Discussion and Applications

We note that, if we disregard the aspects of commitment malleability (see Definition 5.7.2), the compilers for functional composition are more general than those for conjunction. Specifically, we could think of the function f as computing the relation $\mathcal{R}_A$ and passing the witness, unchanged, to the next function g, which in turn computes the relation $\mathcal{R}_B$.

We chose to present two distinct types of compilation (conjunctions and functional compositions) because this approach arguably makes it easier to present our results. Additionally, the simpler compiler (for conjunction) allows us to handle the commitment malleability aspects more directly.

In terms of assumptions, the third and fourth results trade the (additional) efficient witness sampleability property (see Definition 5.7.3) for weaker assumptions on the security of the arguments of knowledge. While the assumption of efficient witness sampleability might seem strong, for functional composition, we can omit this assumption by requiring a structural property on f. This is another reason why we include the fourth result, even though in the following discussion on zkVM in Section 5.8, we only require the compilers for conjunction.

5.8 Simulation extractability of zkVMs

5.8.1 Preliminaries on SNARK VMs

Here we provide an abstract treatment of virtual machines. We start from this general definition:

Definition 5.8.1 (Instruction Set (Execution))**.** *Let $\gamma, k \in \mathbb{N}$. An instruction set for a virtual machine with k registers and codewords of size γ is an efficiently computable function* $\mathsf{Execute} : \{0,1\}^{\gamma\cdot(k+4)} \to \{0,1\}^{\gamma\cdot k}$.

We want to describe the relation which describes a virtual machine execution. Consider the circuit in Fig. 5.18. This is parameterized by an *instruction set* $\mathsf{Execute}$, an execution bound t, a bound on the number of register k, codewords of size γ, and a bound on the output size o. We denote the circuit thus parameterized as $\mathtt{VM}_{\mathsf{Execute},t,o}$. (For simplicity, we hide all the parameters but $\mathsf{Execute}$, and simply write $\mathtt{VM}_{\mathsf{Execute}}$ whenever the parameters are clear from the context.) Following [AST24], we define the commit-and-prove relation:

$$\mathcal{R}^{\mathsf{Execute}}_{\mathsf{zkVM}}((t,o),(\mathsf{P}_{\mathsf{code}},\mathbb{x},\mathbb{y}),\mathbb{z}) \iff \mathtt{VM}_{\mathsf{Execute}}(\mathsf{P}_{\mathsf{code}},\mathbb{x},\mathbb{z}) = \mathbb{y} \qquad (\dagger)$$

The virtual machine $\mathtt{VM}_{\mathsf{Execute}}(\mathsf{P}_{\mathsf{code}}, \mathbb{x}, \mathbb{z})$:

Set $(\mathsf{sregs}, \mathsf{regs}) \leftarrow 0^{k+4}$, $\mathsf{mem} \leftarrow \mathbb{x} \| \mathbb{z}$.
Iterate for t times the following:

- *Update Program-Counter:* $\mathsf{sregs}[0] \leftarrow \mathsf{regs}[0]$.
- *Fetch:* $\mathsf{sregs}[1] \leftarrow \mathsf{P}_{\mathsf{code}}[\mathsf{sregs}[0]]$.
- *Read-and-Write Operations:*
  - $\mathsf{sregs}[2] \leftarrow \mathsf{mem}[\mathsf{regs}[1]]$, //read from memory
  - $\mathsf{sregs}[3] \leftarrow \mathsf{regs}[3]$, //load to special register
  - $\mathsf{mem}[\mathsf{regs}[2]] \leftarrow \mathsf{sregs}[3]$, //write to memory
- *Execute:* $\mathsf{regs} \leftarrow \mathsf{Execute}(\mathsf{sregs})$

Output $\mathbb{y} = \mathsf{mem}[0 : o]$.

Figure 5.18: The VM with parameters the instruction set Execute and a time bound t. The inputs are the program code $\mathsf{P}_{\mathsf{code}}$, a public input $\mathbb{x}$ and a private input $\mathbb{z}$, the output of the VM is $\mathbb{y}$. The machines load on the memory the inputs and executes t steps, the output $\mathbb{y}$ of the machine is the state of the memory after t steps. There are four *special* registers: sregs[0] stores the current program counter, sregs[1] stores the next instruction, while sregs[1] and sregs[2] store the (two) operands for the next instruction, in particular, sregs[1] stores data fetched from the main memory and sregs[2] stores data from the result of the previous instruction. The instructions in Execute do not change the content of the special registers and update the program counter for the fetch of the next instruction in regs[0]. The VM, at any iteration, writes in memory at location regs[2] the content of sregs[3] and at sregs[3] the content of regs[3], these are (somewhat arbitrary) operations to allow flow of information from regs to sregs and from sregs to memory: notice that different architectures performing additional reading/writing operations are theoretically (and practically) equivalent.

5.8.1.1 Splitting $\mathcal{R}_{\mathsf{zkVM}}$ in its logical components.

We now show how to approach proving the relation $\mathcal{R}_{\mathsf{zkVM}}$ from a modular perspective. The way we "split" relation $\mathcal{R}_{\mathsf{zkVM}}$ will roughly follow the lookup-singularity approach in [AST24]. For this reason we will isolate an execution component (which in [AST24] is performed through the lookup argument Lasso) and "anything else" (in relation $\mathcal{R}^{\star}$) roughly consisting of instruction fetching (which we abstracted out in our VM model) and memory checking. For simplicity, we do not break this second part further; our goal is to showcase the modular flavor of zkVMs and to provide a blueprint that can be specialized in follow-up works[12]. We thus define the commit-and-prove relation below:

$$\mathcal{R}_{\mathsf{Execute}}(\mathbb{w}_{\mathsf{regs}}, \mathbb{w}_{\mathsf{sregs}}) \iff \forall i \in [t-1] : \mathsf{Execute}(\mathbb{w}_{\mathsf{sregs}}[i]) = \mathbb{w}_{\mathsf{regs}}[i+1]$$

Here $(\mathbb{w}_{\mathsf{sregs}}[i], \mathbb{w}_{\mathsf{regs}}[i])$ is the state of the registers of the virtual machine at the i-th step of computation. Looking ahead, we associate the tuple $(\mathbb{w}_{\mathsf{sregs}}, \mathbb{w}_{\mathsf{regs}})$ with the trace of the virtual machine in the computation of the program $\mathsf{P}_{\mathsf{code}}$ on input $(\mathbb{x}, \mathbb{z})$.

[12]The work in [AST24] actually logically separates memory checking and instruction fetching. Both the components they use for these modules can be thought of more or less specialized versions of Spartan. Therefore, in spirit, our the instantiations we present in Section 5.8.4 are still applicable to the original presentation in [AST24].

Definition 5.8.2. *The* zkVM-complementary relation *is the relation* $\mathcal{R}^\star$ *such that for any execution set* $\mathsf{Execute}$*, and for any input* $(\mathsf{P}_{\mathsf{code}}, \mathbb{x}, \mathbb{y}), \mathbb{z}$*:*

$$\mathcal{R}^{\mathsf{Execute}}_{\mathsf{zkVM}}((\mathsf{P}_{\mathsf{code}}, \mathbb{x}, \mathbb{y}), \mathbb{z}) \iff \exists(\mathbb{w}_{\mathsf{regs}}, \mathbb{w}_{\mathsf{sregs}}, \mathbb{w}_{\mathsf{mem}}) : \mathcal{R}_{\mathsf{Execute}}(\mathbb{w}_{\mathsf{regs}}, \mathbb{w}_{\mathsf{sregs}}) \wedge \mathcal{R}^\star(\mathsf{P}_{\mathsf{code}}, \mathbb{x}, \mathbb{y}, (\mathbb{w}_{\mathsf{regs}}, \mathbb{w}_{\mathsf{sregs}}, \mathbb{w}_{\mathsf{mem}}))$$

where $\mathcal{R}^{\mathsf{Execute}}_{\mathsf{zkVM}}$ *is the relation defined as in Eq.* (†)*.*

Intuitively, the relation $\mathcal{R}^\star$ needs to handle the logic of the virtual machine and make sure that the memory accesses, during the execution of the program, are consistent (namely, that we read the correct instructions from $\mathsf{P}_{\mathsf{code}}$, we perform the read and write operations, and that if the virtual machine reads from the memory the value v at location i, it means that the last time the virtual machine wrote at location i, it wrote the value v).

5.8.2 A General Theorem on the Non-Malleability of SNARK VMs

We say that a commit-and-prove argument of knowledge for $\mathcal{R}^\star$ has *separate commitments* (for CS) if the witnesses $\mathbb{w}_{\mathsf{regs}}, \mathbb{w}_{\mathsf{sregs}}$ and $\mathbb{w}_{\mathsf{mem}}$ are committed separately. Namely, the witness $\mathbb{w} := (\mathbb{w}_{\mathsf{regs}}, \mathbb{w}_{\mathsf{sregs}}, \mathbb{w}_{\mathsf{mem}})$ for $\mathcal{R}^\star$ is committed as $\mathsf{c}_X, \rho_X \leftarrow \mathsf{Com}(\mathsf{ck}, \mathbb{w}_X)$ for $X \in \{\mathsf{regs}, \mathsf{sregs}, \mathsf{mem}\}$ and $\mathsf{c} := (\mathsf{c}_{\mathsf{regs}}, \mathsf{c}_{\mathsf{sregs}}, \mathsf{c}_{\mathsf{mem}})$.

Theorem 5.8.1. *For any instruction set* $\mathsf{Execute}$*, let* Π *be a zero-knowledge argument of knowledge for* $\mathcal{R}_{\mathsf{Execute}}$ *that is simulation extractable, and let* $\Pi^\star$ *be an argument of knowledge for* $\mathcal{R}^\star$ *that has separate commitments. There exists a simulation-extractable zkVM if one of the following holds:*

1. $\Pi^\star$ *is simulation-extractable and zero-knowledge.*
2. $\Pi^\star$ *is witness-hiding and* Π *is a signature-of-knowledge.*

The theorem follows as an application of Theorem 5.7.1. The interesting case is when $\Pi^\star$ is WI. In this case, we additionally need to prove efficient witness sampleability by showing an *altered* instruction set that simply prints the output $\mathbb{y}$ into memory.

Proof. The theorem follows as an application of Theorem 5.7.1 and in particular of the first statement assuming the conditions in item (1) and third statement assuming the condition in item (2). We start with the more interesting case where we combine a knowledge-sound and witness-hiding scheme for $\mathcal{R}^\star$ with a simulation-extractable scheme for $\mathcal{R}_{\mathsf{Execute}}$. We can assume that the witness $\mathbb{w} := (\mathbb{w}_{\mathsf{regs}}, \mathbb{w}_{\mathsf{sregs}}, \mathbb{w}_{\mathsf{mem}})$ for $\mathcal{R}^\star$ is committed separately, namely $\mathsf{c}_X, \rho_X \leftarrow \mathsf{Com}(\mathsf{ck}, \mathbb{w}_X)$ for $X \in \{\mathsf{regs}, \mathsf{sregs}, \mathsf{mem}\}$ and $\mathsf{c} := (\mathsf{c}_{\mathsf{regs}}, \mathsf{c}_{\mathsf{sregs}}, \mathsf{c}_{\mathsf{mem}})$.

We define the functions $M_c(\mathsf{c}_{\mathsf{sregs}}, \mathsf{c}_{\mathsf{regs}}, \mathsf{c}_{\mathsf{mem}}) = (\mathsf{c}_{\mathsf{sregs}}, \mathsf{c}_{\mathsf{regs}})$, and similarly, $M(\mathbb{w}) = (\mathbb{w}_{\mathsf{sregs}}, \mathbb{w}_{\mathsf{regs}})$ and $M_\rho(\rho_{\mathsf{sregs}}, \rho_{\mathsf{regs}}, \rho_{\mathsf{mem}}) = (\rho_{\mathsf{sregs}}, \rho_{\mathsf{regs}})$. It is trivial to show that the commitment for $\Pi^\star$ is M-malleable and $M(\mathbb{w})$ is a witness for $\mathcal{R}_{\mathsf{Execute}}$ as required by the third statement of Theorem 5.7.1. We define the simulation-extractable zkVM as the composition $\bar{\Pi}_\wedge$ in Theorem 5.7.1 with M-malleable commitment between Π and $\Pi^\star$. We need to show that $\mathcal{R}^\star$ is efficiently witness sampleable and always satisfiable. For the former, consider the following "altered" virtual machine.

The altered virtual machine $\mathtt{VM}'_{\mathsf{Execute}}$:

- Run for $(t-o)$ iterations the code of $\mathtt{VM}_{\mathsf{Execute}}(\mathsf{P}_{\mathsf{code}}, \mathbb{x}, \mathbb{z})$ with $\boxed{\mathbb{z} \leftarrow \bar{0}}$.
- For $i \in [o]$ runs the following:
  - *Update Program-Counter:* $\mathsf{sregs}[0] \leftarrow \mathsf{regs}[0]$.
  - *Fetch:* $\mathsf{sregs}[1] \leftarrow \mathsf{P}_{\mathsf{code}}[\mathsf{sregs}[0]]$.
  - *Read-and-Write Operations:*
    - $\mathsf{sregs}[2] \leftarrow \mathsf{mem}[\mathsf{regs}[1]]$, //read from memory
    - $\mathsf{sregs}[3] \leftarrow \mathsf{regs}[3]$, //load to special register
    - $\mathsf{mem}[\mathsf{regs}[2]] \leftarrow \mathsf{sregs}[3]$, //write to memory
  - *Execute:* $\boxed{\mathsf{regs} \leftarrow (0, 0, i, \mathbb{y}_i, \bar{0})}$
- Output $\mathbb{y} = \mathsf{mem}[0 : o]$.

We can compute the trace $\mathbb{w} := (\mathbb{w}_{\mathsf{sregs}}, \mathbb{w}_{\mathsf{regs}}, \mathbb{w}_{\mathsf{mem}})$ associated with the execution of the altered virtual machine $\mathtt{VM}'_{\mathsf{Execute}}$. In the code of $\mathtt{VM}'_{\mathsf{Execute}}$, the only difference with respect to an execution of $\mathtt{VM}_{\mathsf{Execute}}(\mathsf{P}_{\mathsf{code}}, \mathbb{x}, \bar{0})$ is that at the end we force to write to the first o locations of the memory the value $\mathbb{y}$, thus forcing the output of $\mathtt{VM}'_{\mathsf{Execute}}$ to $\mathbb{y}$. Notice that $\mathbb{w}$ is a valid witness for $\mathcal{R}^\star$ on instance $(\mathsf{P}_{\mathsf{code}}, \mathbb{x}, \mathbb{y})$, this is because $\mathcal{R}^\star$ does not enforce the consistency of the registers regs between two consecutive steps and, in particular, during the last o iterations.

Additionally, we notice that, if the commitment scheme is perfectly hiding, then language $\mathcal{L}_{\hat{\mathcal{R}}^\star}$ is always satisfiable because we can execute the procedure above to create a valid witness $\mathbb{w}$ and, although inefficiently, we can always find ρ such that the commitment opens to $\mathbb{w}$ with opening ρ.

If we assume (2) we can define the simulation-extractable zkVM as the composition in Theorem 5.7.1 between Π and $\Pi^\star$. □

5.8.3 Signature-of-Knowledge with delayed message

The previous theorem highlights that, in many scenarios, we can obtain simulation extractability even when one of the components of the composed argument is malleable. However, the caveat is that we need to require that the second component is not only simulation extractable but also a signature of knowledge. It is rather easy to instantiate a signature of knowledge from an FS-based simulation extractable argument of knowledge, by including the message to the hashed view. However, there is an efficiency bottleneck in doing so in our compilers from Theorem 5.7.1. In fact, for example in the third statement, the message contains the proofs π_A, which enforce a sequentiality in the proofs' generation by the prover, namely π_A needs to be generated before π_B.

To mitigate such a bottleneck, we describe a notion of signature of knowledge where, roughly speaking, the message can be fed at the very end of the prover's computations. We refer to this as a *signature of knowledge with delayed message*. Informally, the prover's algorithm $\mathcal{P}$ can be divided into two procedures $\mathcal{P}_1$ and $\mathcal{P}_2$: the first procedure $\mathcal{P}_1$ takes as input the instance and witness (thus it is independent of the message), while $\mathcal{P}_2$ receives the internal state of $\mathcal{P}_1$ and the message, namely $\mathcal{P}(\mathsf{pp}, \mathtt{msg}, \mathbb{x}, \mathbb{w}) = \mathcal{P}_2(\mathtt{msg}, \mathcal{P}_1(\mathsf{pp}, \mathbb{x}, \mathbb{w}))$. The efficiency property we

are interested in is that non-trivially $t(\mathcal{P}_2) < t(\mathcal{P}_1)$ where, very roughly speaking, $t(A)$ is the computational complexity of the algorithm A.

Fiat-Shamir-based Approach. We show that in Fiat-Shamir-based signature-of-knowledge the message does not need to be hashed until the round k where k-zero-knowledge and k-unique-response hold. This might enable for delayed message when the index k is the last (or more generally, when all the commitments have been computed and sent).

Theorem 5.8.2. *Let Π be a $(2r+1)$-message public-coin interactive argument. Let $\Pi_{\mathsf{FS}^*,k}$ be the Fiat-Shamir transform where the k-th challenge is derived as $\mathsf{H}(\mathsf{pp}, \mathtt{msg}, \mathbb{x}, \pi|_k)$ for an input message $\mathtt{msg}$. If there is $k \in [r]$ such that $\Pi_{\mathsf{FS}^*,k}$ satisfies knowledge-soundness, k-zero-knowledge and k-unique response, then $\Pi_{\mathsf{FS}^*,k}$ is a signature of knowledge.*

Sketch. The proof proceeds exactly as Theorem BLA in [DG23]. In particular, we can define a knowledge-sound adversary $\mathcal{B}$ for $\Pi_{\mathsf{FS}^*,k}$ from the sim-ext adversary $\mathcal{A}$ for $\Pi_{\mathsf{FS}^*,k}$ by internally program the random oracle only on the input defined by the k-th round when running the zero-knowledge simulator and reply all the other queries using the random oracle interface.

Eventually $\mathcal{A}$ outputs its forgery. Such a forgery is considered valid for $\mathcal{B}$ if the verifier does not need to query the random oracle at the input programmed by $\mathcal{B}$ when verifying the forgery. When such an event happens we say that $\tilde{\pi}$ contains a *critical* RO-query.

Let $(\tilde{\mathtt{msg}}, \tilde{\mathbb{x}}, \tilde{\pi})$ be the forgery of $\mathcal{A}$ and $(\tilde{\mathtt{msg}}, \tilde{\mathbb{x}}, \tilde{\pi}) \notin \mathcal{Q}_{\mathsf{sim}}$, we can proceed with a case analysis:

- If $(*, \tilde{\mathbb{x}}, *) \notin \mathcal{Q}_{\mathsf{sim}}$ then $\tilde{\pi}$ does not contain any critical queries, and we can reduce directly to the knowledge soundness.
- Otherwise, if $\tilde{\pi}_{|k} = \pi_{|k}$ and $\tilde{\mathtt{msg}} \neq \mathtt{msg}$ for simulated $(\mathtt{msg}, \tilde{\mathbb{x}}, \pi)$, then we can break k-UR since $\mathsf{H}(\tilde{\mathtt{msg}}, \pi_{|k}) \neq \mathsf{H}(\mathtt{msg}, \pi_{|k})$ with overwhelming probability.
- Finally, if $(\tilde{\mathtt{msg}}, \tilde{\mathbb{x}}, *) \in \mathcal{Q}_{\mathsf{sim}}$ but $\tilde{\pi}_{|k} \neq \pi_{|k}$ then $\tilde{\pi}$ does not contain any critical queries, and we can reduce directly to the knowledge soundness.

□

Black-box approach. A second, black-box approach is based on a technique (which we believe to be folklore) relying on one-time signature.

Definition 5.8.3. *We say that $\Sigma = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ is a one-time signature if:*

Syntax. *The three algorithms are PPT where $\mathsf{KGen}(1^\lambda)$ returns a pair pk, sk of public and secret keys, $\mathsf{Sign}(\mathsf{sk}, \mathtt{msg})$ with $\mathtt{msg} \in \{0,1\}^\lambda$ returns a signature $\sigma_{\mathtt{msg}}$, and $\mathsf{Verify}(\mathsf{pk}, \mathtt{msg}, \sigma)$ returns a decision bit.*

Correctness. *For any $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{KGen}(1^\lambda)$ and any $\mathtt{msg} \in \{0,1\}^\lambda$ we have:*

$$\mathsf{Verify}(\mathsf{pk}, \mathtt{msg}, \mathsf{Sign}(ss, \mathtt{msg})) = 1$$

One-time unforgeability. *For any PT adversary $\mathcal{A}$ that upon input the public key and an adaptively chosen message $\mathtt{msg}$ (and a signature σ for it) outputs $(\tilde{\mathtt{msg}}, \tilde{\sigma})$, with $(\tilde{\mathtt{msg}}, \tilde{\sigma}) \neq (\mathtt{msg}, \sigma)$:*

$$\Pr\Big[\mathsf{Verify}(\mathsf{pk}, \mathcal{A}(\mathsf{pk}, \mathsf{Sign}(\mathsf{sk}, \mathtt{msg}))) = 1 : (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)\Big] \in \mathsf{negl}(\lambda)$$

Let Π be a signature of knowledge for $\mathcal{R}$ and Σ a one-time signature, consider the protocol $\Pi' \coloneqq (\mathcal{P}, \mathcal{V})$ for $\mathcal{R}$ where:

- $\mathcal{P}(\mathsf{pp}, \mathtt{msg}, \mathbb{x}, \mathbb{w})$ samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$, computes $\pi \leftarrow \Pi.\mathcal{P}(\mathsf{pp}, \mathsf{pk}, \mathbb{x}, \mathbb{w})$ and $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathtt{msg} \| \mathbb{x})$, returns $\pi' \coloneqq (\pi, \mathsf{pk}, \sigma)$.
- $\mathcal{V}(\mathsf{pp}, \mathtt{msg}, \mathbb{x}, \pi')$ returns $\Pi.\mathcal{V}(\mathsf{pk}, \mathbb{x}, \pi)$ and $\mathsf{Verify}(\mathsf{pp}, \mathsf{pk}, \mathbb{x}, \sigma)$.

Theorem 5.8.3. *If Π is a signature-of-knowledge for $\mathcal{R}$ and Σ is a one-time signature then Π' is a signature-of-knowledge for $\mathcal{R}$.*

sketch. The event that there exist two simulation queries that have the same public key for the one-time signature scheme is negligible, as otherwise we can break one-time unforgeability. Let $(\tilde{\mathtt{msg}}, \tilde{\mathbb{x}}, (\tilde{\mathsf{pk}}, \tilde{\pi}, \tilde{\sigma}))$ be the forgery of the adversary. If $\tilde{\mathsf{pk}}$ is not fresh, i.e., there exists a simulated proof $(\mathsf{pk}, \pi, \sigma)$ such that $\mathsf{pk} = \tilde{\mathsf{pk}}$, then it must be $(\tilde{\mathtt{msg}}, \tilde{\sigma}) = (\mathtt{msg}, \sigma)$, as otherwise we break one-time unforgeability, but then $(\tilde{\mathbb{x}}, \tilde{\pi}) \neq (\mathbb{x}, \pi)$ which implies that $(\tilde{\mathtt{msg}}, \tilde{\mathbb{x}}, \tilde{\pi})$ is a valid forgery for the inner-scheme Π. On the other hand, when $\tilde{\mathsf{pk}}$ is fresh, $(\tilde{\mathsf{pk}}, \tilde{\mathbb{x}}, \tilde{\pi})$ is a valid forgery for Π independently of the signed message $\tilde{\mathtt{msg}}$. □

5.8.4 The Lookup-Singularity is Non-Malleable

As already mentioned in this section, we can realize an argument of knowledge for $\mathcal{R}_{\mathsf{Execute}}$ using a lookup argument. The basic idea is to consider the truth table of the instruction set $\mathsf{Execute}$ as the table, and the execution trace $\mathbb{w}_{\mathsf{Execute}}$ as the subvector. Although the truth table of the instruction set $\mathsf{Execute}$ is exponentially large, [AST24] shows that the truth table for the instruction set of RISC-V is decomposable (Definition 5.6.1).

Below we use the concept that an instruction set is decomposable if it can be described by a decomposable table (Definition 5.6.1).

Theorem 5.8.4. *If $\mathsf{Execute}$ is a decomposable instruction set, then zkLasso (see Section 5.6) implies a simulation-extractable argument of knowledge and a signature of knowledge with delayed message for $\mathcal{R}_{\mathsf{Execute}}$.*

Proof. Fixed $\mathsf{Execute}$, we can define the argument system for $\mathcal{R}_{\mathsf{Execute}}$ that runs the prover and verifier of zkLasso with parameter a table E that encodes the truth table of $\mathsf{Execute}$. Namely, E is the table such that $E[\mathsf{sregs}] = \mathsf{Execute}(\mathsf{sregs})$ for any $\mathsf{sregs} \in \{0,1\}^{3\cdot\gamma}$. Recall that the truth table of $\mathsf{Execute}$, namely E, is decomposable. To prove $\mathcal{R}_{\mathsf{Execute}}(\mathbb{w}_{\mathsf{sregs}}, \mathbb{w}_{\mathsf{regs}})$ we prove that $\mathcal{R}_{\mathsf{lookup}}(E, \mathbb{w}_{\mathsf{sregs}}, \mathbb{w}_{\mathsf{regs}})$ where $\mathbb{w}_{\mathsf{sregs}}$ defines the committed indexes and $\mathbb{w}_{\mathsf{regs}}$ the committed sub-table.

Additionally, we notice that Theorem 5.6.1 and Theorem 5.8.2 imply we can apply the FS-transform to zkLasso to create a signature-of-knowledge with delayed messages and thus a signature-of-knowledge with delayed messages for $\mathcal{R}_{\mathsf{Execute}}$. □

Definition 5.8.4 (Joltish)**.** *Let $\mathsf{Execute}$ be a decomposable instruction set and let $\Pi^\star$ be an argument of knowledge for $\mathcal{R}^\star$. We call Joltish (instantiated with $\Pi^\star$) the argument for $\mathcal{R}_{\mathsf{zkVM}}^{\mathsf{Execute}}$ derived from the one of the compilers for conjunction in Fig. 5.17 and Theorem 5.8.1 where Π_{Execute} for $\mathcal{R}_{\mathsf{Execute}}$ is zkLasso.*

5.8.4.1 An efficient SIM-EXT zkVM for RISC-V

Following [AST24], and as a corollary of Theorems 5.6.1 and 5.8.1, we have the following:

Corollary 5.8.1. *Let* Execute *be a decomposable instruction set, then there exists* $\Pi^\star$ *as in Definition 5.8.4 s.t.* Joltish *instantiated with* $\Pi^\star$ *is simulation-extractable zkVMs for* $\mathcal{R}_{\mathsf{zkVM}}$ *yielded by* Execute.

To argue that Jolt, or more precisely its zero-knowledge version, is simulation-extractable, it remains to show that the hypotheses of Theorem 5.8.1 hold for Jolt's implementation of the argument of knowledge for $\mathcal{R}^\star$.

In detail, in [AST24], Arun, Setty, and Thaler show how to realize a succinct argument of knowledge for $\mathcal{R}^\star$ using a commit-and-prove argument of knowledge for R1CS (they use Spartan [Set20]) and a commit-and-prove argument of knowledge for memory consistency based on the grand-product argument and memory checking techniques from [BEG+91].

More specifically, the latter parses $\mathbb{w}_{\mathsf{mem}}$ as a list of memory operations of the form (M, τ, o, l, v), where $M \in \{\mathsf{P}_{\mathsf{code}}, \mathsf{mem}\}$ indicates which of the memories[13] to read from or write to, τ is a timestamp, o is the operation (e.g., read or write), l is a location, and v is a value. The former proves that, assuming the memory accesses are consistent, the logic of the virtual machine is executed correctly; namely, the fetch and read-and-write operations (on the registers) are executed and iterated t times.

In Fig. 5.14, we show a zero-knowledge variant of the grand-product argument, which allows us to state that the sub-scheme for $\mathcal{R}^\star$ in Joltish is both knowledge-sound and zero-knowledge, thus enabling us to use the result from our theorem Theorem 5.7.1.

[13] The $\mathsf{P}_{\mathsf{code}}$ is a read-only memory, thus additional optimizations are available, while mem is a read-and-write memory.

Chapter 6

Mix-Nets from Re-Randomizable PKE

This chapter is extracted from "Mix-Nets from Re-Randomizable and Replayable CCA-secure Public-Key Encryption", published in SCN 2022.

6.1 Introduction

Mixing Networks (aka mix-nets), originally proposed by Chaum [Cha81], are protocols that allow a set of senders to send messages anonymously. Typically, a mix-net is realized by a chain of mix-servers (aka mixers) that work as follows. Senders encrypt their messages and send the ciphertexts to the first mix-server in the chain; each mix-server applies a transformation to every ciphertext (e.g., partial decryption, or re-encryption), re-orders the ciphertexts according to a secret random permutation, and passes the new list to the next mix-server. The idea is that the list returned by the last mixer contains (either in clear or encrypted form, depending on the mixing approach) the messages sent by the senders in a randomly permuted order.

Mix-net protocols are fundamental building blocks to achieve privacy in a variety of application scenarios, including anonymous e-mail [Cha81], anonymous payments [JM99], and electronic voting [Cha81]. Informally, the basic security property of mix-nets asks that, when enough mix-servers are honest, the privacy of the senders of the messages (i.e., "who sent what") is preserved. In several applications, it is also desirable to achieve correctness even in the presence of an arbitrary number of dishonest mixers. This is for example fundamental in electronic voting where a dishonest mixer could replace all the ciphertexts with encrypted votes for the desired candidate.

Realizing Mix-Nets. A popular design paradigm of mixing networks are *re-encryption mix-nets* [PIK94] in which each server decrypts and freshly encrypts every ciphertext. Interestingly, such a transformation can be computed even publicly using re-randomizable encryption schemes (e.g., ElGamal). The process of re-randomizing and randomly permuting ciphertexts is typically called a *shuffle*. Although shuffle-based mix-nets achieve privacy when all the mix-servers behave honestly, they become insecure if one or more mixers do not follow the protocol. An elegant approach proposed to solve this problem is to let each mixer prove the correctness of its shuffle with a zero-knowledge proof. This idea inspired a long series of works on zero-knowledge shuffle arguments, e.g., [BG12, FS01, Gro03, Gro10b, Nef01, TW10, Wik05, Wik09]. Notably, some recent works [BG12, TW10, Wik09] improved significantly over the early solutions, and they have been implemented and tested in real-world applications (elections) [Wik10]. In spite

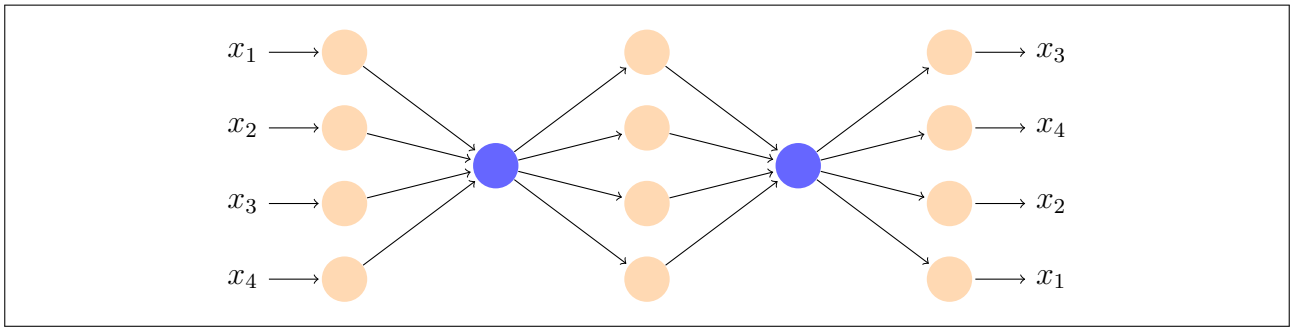


Figure 6.1: Example of MixNet with 4 senders and 2 mixers.

of the last results, zero-knowledge shuffle arguments are still a major source of inefficiency in mix-nets. This is especially a concern in applications like electronic voting where mix-nets need to be able to scale up to millions of senders (i.e., voters).

Mix-Nets from Replayable CCA Security. Most of the research effort for improving the efficiency of mix-nets has been so far devoted to improving the efficiency of shuffle arguments. A notable exception is the work of Faonio *et al.* [FFHR19]. Typical mixing networks based on re-randomizable encryption schemes make use of public-key encryption (PKE) schemes that are secure against chosen-plaintext attack (CPA), thus to obtain security against malicious mixers they leverage on the strong integrity property offered by the zero-knowledge shuffle arguments. The work of Faonio *et al.* instead showed that, by requiring stronger security properties from the re-randomizable encryption scheme, the NP-relation proved by the zero-knowledge shuffle arguments can be relaxed. This design enables faster and more efficient instantiations for the zero-knowledge proof but, on the other hand, requires more complex ciphertexts and thus a re-randomization procedure that is slower in comparison, for example, with the re-randomization procedure for ElGamal ciphertexts. More in detail, Faonio *et al.* propose a secure mixing network in the universal composability model of Canetti [Can01] based on re-randomizable PKE schemes that are replayable-CCA (RCCA) secure (as defined by Canetti *et al.* [CKN03]) and publicly-verifiable. The first notion, namely RCCA security, is a relaxation of the standard notion of chosen-ciphertext security. This notion offers security against malleability attacks on the encrypted message (i.e. an attacker cannot *transform* a ciphertext of a message $\mathtt{msg}$ to a ciphertext of a message $\mathtt{msg}'$) but it still allows for malleability on the ciphertext (i.e. we can re-randomize the ciphertexts). The second requirement, namely public verifiability, requires that anyone in possession of the public key can check that a ciphertext decrypts correctly to a valid message, in other words, that the decryption procedure would not output an error message on input such a ciphertext. Unfortunately, this second requirement is the source of the major inefficiency in the mixing networks of Faonio *et al.*. For example, the re-randomization procedure of the state-of-art non publicly-verifiable re-randomizable PKE scheme with RCCA-security (Rand-RCCA PKE, in brief) in the random oracle model of Faonio and Fiore [FF20] costs 19 exponentiations in a pairing-free cryptographic group, while the re-randomization procedure of the publicly-verifiable Rand-RCCA PKE of [FFHR19] costs around 90 exponentiations plus 5 pairing operations.

6.1.1 Our Contributions

We revisit the mix-net design of Faonio *et al.* [FFHR19]. Our contributions are two-fold: we generalize the mix-net protocol of [FFHR19] showing how to get rid of the cumbersome public verifiability property, and we give a more efficient instantiation for the mix-net protocol based on the (non publicly-verifiable) Rand-RCCA scheme of [FFHR19]. Our generalization of the mix-net protocol is based on two main ideas. The first idea is that, although the verification of the ciphertexts is still necessary, it is not critical for the verification to be public and non-interactive. In particular, we can replace the public verifiability property with a multi-party protocol (that we call a *verify-then-decrypt* protocol) that verifies the ciphertexts before the decryption phase and that decrypts the ciphertexts from the last mixer in the chain only if the verification succeeded. The second idea is that in the design of the *verify-then-decrypt* multiparty protocol we can trade efficiency for security. In particular, we could design a protocol that eventually leaks partial information about the secret key and, if the Rand-RCCA PKE scheme is resilient against this partial leakage of the secret key, we could still obtain a secure mix-net protocol. Along the way, we additionally (1) abstract the necessary properties required by the zero-knowledge proof that the mixers need to attach to their shuffled ciphertexts and (2) give a more careful security analysis of the mixnet protocol. More technically, we define the notion *sumcheck-admissible relation* w.r.t. the Rand-RCCA PKE scheme (see Definition 6.5.1) which is a property of an NP-relation that, informally, states that given two lists of ciphertexts if all the ciphertexts in the lists decrypt to valid messages, then the sum of the messages in the first list is equal to the sum of the messages in the second list. For example, a shuffle relation is a sumcheck-admissible relation, however simpler (and easier to realize in zero-knowledge) NP-relations over the lists of ciphertexts can be considered as well.

Our second contribution is a concrete instantiation of the mix-net protocol. The main idea of our concrete protocol is that many (R)CCA PKE schemes can be conceptually divided into two main components: the first *"CPA-secure"* component assures that the messages are kept private, while the second component assures the integrity of the ciphertexts, namely, the component can identify malformed ciphertexts and avoid dangerous decryptions through the CPA-secure component. Typical examples for such PKE schemes are those based on the Cramer-Shoup paradigm [CS02]. Intuitively, these schemes should be secure even if the adversary gets to see the secret key associated with the second component under the constraint that once such leakage is available the adversary must lose access to the decryption oracle. This suggests a very efficient design for the *verify-then-decrypt* multiparty protocol: the mixers commit to secret shares of the secret key, once all the ciphertexts are available the mixers open to the secret key material for the second component, now any mixer can non-interactively and efficiently verify the validity of the ciphertexts. If all the ciphertexts are valid the mixers can engage a CPA-decryption multiparty protocol for the ciphertexts in the last list. As last contribution, we show that the Rand-RCCA PKE scheme of [FFHR19] is leakage resilient (under the aforementioned notion) and we instantiate all the necessary parts.

A final remark, an important property of a mixnet protocol is the so-called *auditability*[1], namely the capability of an external party to verify that a given transcript of a protocol execution has produced an alleged output. Intuitively, mixnets based on non-interactive zero-knowledge proofs of shuffle usually should have this property. However, one must be careful,

[1]This notion is sometimes called *verifiability*, however, we prefer to use the term "auditability" to avoid confusion with the verifiability of the ciphertexts.

because not only the shuffling phase, but the full mixnet protocol should be auditable. In particular, for our mixnet protocol to be auditable the verify-then-decrypt protocol should be auditable as well. We show that the latter protocol for our concrete instantiation is indeed auditable.

6.2 All-but-One tag-based NIZK systems

Let $\mathsf{NIZK} = (\mathsf{Init}, \mathsf{P}, \mathsf{V})$ be a NIZK proof system for a relation $\mathcal{R}$ with tag space $\mathcal{T}$, and let $\mathsf{TPInit}(\mathsf{pp}, \tau)$ be an algorithm that upon input pp and a tag $\tau \in \mathcal{T}$ outputs a common reference string crs and trapdoor information $(\mathsf{tpe}, \mathsf{tps})$. Let $\tau \in \mathcal{T}$, and $\mathcal{T}'$ subset of $\mathcal{T}$. We define the following three properties:

- We say that TPInit is CRS indistinguishable w.r.t. NIZK if the common reference string generated by $\mathsf{Init}(\mathsf{pp})$ and the one generated by $\mathsf{TPInit}(\mathsf{pp}, \tau)$ are computationally indistinguishable, i.e. if for any sequences of $\{\mathsf{pp}_\lambda \leftarrow \mathsf{Setup}(1^\lambda)\}_{\lambda \in \mathbb{N}}$, and for any PT adversary $\mathcal{A}$:

$$|\Pr[\mathcal{A}(\mathsf{crs}) = 1 : \mathsf{crs} \leftarrow\!\!\$\ \mathsf{Init}(\mathsf{pp}_\lambda)] - \Pr[\mathcal{A}(\mathsf{crs}) = 1 : \mathsf{crs}, \mathsf{tpe}, \mathsf{tps} \leftarrow\!\!\$\ \mathsf{TPInit}(\mathsf{pp}_\lambda)]| \in \mathsf{negl}(\lambda)$$

- We say that NIZK is $\mathcal{T}'$-tag Composable Perfect Zero-Knowledge if there exists a PT TPInit that is CRS indistinguishable w.r.t. NIZK and for any pp, τ and any $(\mathsf{crs}, \mathsf{tpe}, \mathsf{tps}) \leftarrow \mathsf{TPInit}(\mathsf{pp}, \tau)$, and any $(x, w) \in \mathcal{R}$ and any $\tau' \in \mathcal{T}'$ we have that the distributions $\mathsf{P}(\mathsf{crs}, \tau', x, w)$ and $\mathsf{Sim}(\mathsf{tps}, \tau', x)$ are equivalently distributed.

- We say that NIZK is $\mathcal{T}'$-tag Adaptive Perfect g-Extractable if there exists a PT TPInit that is CRS indistinguishable w.r.t. NIZK and for any pp, τ there exists a PT extractor $\mathcal{E}$ such that for any pp, τ, for any $(\mathsf{crs}, \mathsf{tps}, \mathsf{tpe}) \leftarrow \mathsf{TPInit}(\mathsf{pp}, \tau)$, and for any (possibly unbounded) adversary $(\tau', x, \pi) \leftarrow \mathcal{A}(\mathsf{crs})$ we have that if $\tau' \in \mathcal{T}'$ and $\mathsf{V}(\tau_{\mathrm{snd}}, x, \pi) = 1$ then $\mathcal{E}(\mathsf{tpe}, \tau_{\mathrm{snd}}, x, \pi)$ outputs z such that $\exists w : (x, w) \in \mathcal{R} \wedge g(w) = z$.

- We say that NIZK is $\mathcal{T}'$-tag Adaptive Perfect Sound if there exists a PT TPInit that is CRS indistinguishable w.r.t. NIZK and for any pp, τ, for any pp, τ, for any $(\mathsf{crs}, \mathsf{tps}, \mathsf{tpe}) \leftarrow \mathsf{TPInit}(\mathsf{pp}, \tau)$, and for any (possibly unbounded) adversary $(\tau', x, \pi) \leftarrow \mathcal{A}(\mathsf{crs})$ we have that if $\tau' \in \mathcal{T}'$ and $\mathsf{V}(\tau_{\mathrm{snd}}, x, \pi) = 1$ then $\exists w : (x, w) \in \mathcal{R}$.

Definition 6.2.1 (All-but-One NIZK)**.** *We say that a tag-based NIZK* NIZK *for a relation* $\mathcal{R}$ *and with tag-space* $\mathcal{T}$ *is:*

- *All-but-one Perfect Sound if for all* $\tau \in \mathcal{L}$ *it is* $\{\tau\}$*-tag Composable Perfect Zero-Knowledge and* $\mathcal{T} \setminus \{\tau\}$*-tag Adaptive Perfect Sound.*

- *All-but-one Perfect Hiding and g-Extractable if for all* $\tau \in \mathcal{L}$ *it is* $\mathcal{T} \setminus \{\tau\}$*-tag Composable Perfect Zero-Knowledge and* $\{\tau\}$*-tag Adaptive Perfect g-Extractable.*

For ABO-NIZK we sometimes use the alias $\mathsf{ABOInit}$ for the algorithm TPInit.

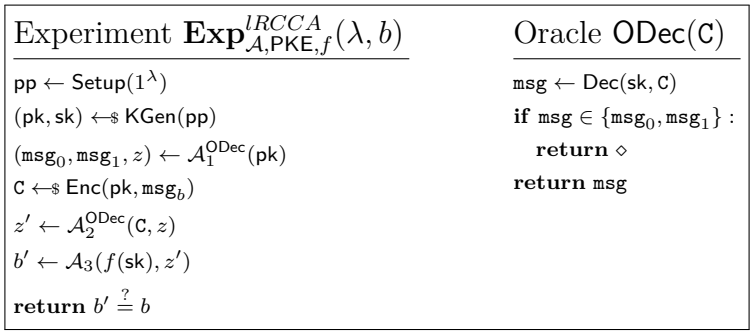


Figure 6.2: The lRCCA security experiment.

Construction of an ABO Perfect Hiding. Consider the instantiation of GS Proof system of [EHK+13] based on $\mathcal{D}_k$-MDDH. The common reference string is of the following two forms:

$$[\vec{A} \| \vec{A}\vec{w}] \qquad \text{Perfect Sound Mode}$$
$$[\vec{A} \| \vec{A}\vec{w} - \vec{z}] \qquad \text{Perfect Hiding Mode}$$

where $\vec{A} \leftarrow\$ \mathcal{D}_k$, $\vec{w} \leftarrow\$ \mathbb{Z}_q^k$ and $\vec{z} \notin span(\vec{A})$ is a fixed and public vector. We can consider a NIZK with tags where the common reference string is made by two independent CRSs $\mathsf{crs}_1, \mathsf{crs}_2$, both the verifier and the prover on input a tag $\tau \in \mathbb{Z}_q$ derive a CRS $\mathsf{crs}_\tau = \mathsf{crs}_1 + \mathsf{crs}_2 \cdot \tau$. We are ready to define the $\mathsf{ABOInit}_{\text{hid}}$.

$\underline{\mathsf{ABOInit}_{\text{hid}}(\mathsf{pp}, \tau^*)}$:

1. Sample $\vec{A}_1, \vec{A}_2$ and $\vec{w}_1, \vec{w}_2$ and set $\mathsf{crs}'_1 = (\vec{A}_1 \| \vec{A}_1\vec{w}_1)$ and $\mathsf{crs}'_2 = (\vec{A}_2 \| \vec{w}_2 - \vec{z})$;
2. Set $\mathsf{crs}_1 = \mathsf{crs}'_1 - \mathsf{crs}'_2 \cdot \tau^*$ and $\mathsf{crs}_2 = \mathsf{crs}'_2$;
3. Output $\mathsf{crs}_1, \mathsf{crs}_2$.

The all-but-one composable zero-knowledge comes readily from the $\mathcal{D}_k$-MDDH assumption and the composable zero-knowledge of GS proofs. The all-but-one adaptive perfect soundness comes readily from the adaptive perfect soundness of GS proofs, in fact we notice that $\mathsf{crs}_{\tau^*} = \mathsf{crs}'_1 - \tau^*\mathsf{crs}'_2 + \tau^*\mathsf{crs}'_2 = \mathsf{crs}'_1$ which allows for perfectly sound proofs.

6.3 Replayable CCA with Leakage Security

We rely on the following notion of security for Rand-PKE. Our notion is similar to the RCCA security game, with the difference that here $\mathcal{A}$ is given the additional leakage $f(\mathsf{sk})$ just before committing to the verdict bit b'. $\mathcal{A}$ cannot invoke the decryption oracle after the leakage occurs.

Definition 6.3.1 (RCCA with leakage Security)**.** *Consider the experiment* $\mathbf{Exp}^{lRCCA}_{\mathcal{A},\mathsf{PKE},f}$ *in Fig. 6.2, with parameters* λ*, an adversary* $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$*, a PKE scheme* PKE*, and a leakage function* f*.* PKE *is* leakage-resilient replayable CCA-secure (lRCCA-secure) *w.r.t. a leakage function* f *if for any PPT adversary* $\mathcal{A}$*:*

$$\mathbf{Adv}^{lRCCA}_{\mathcal{A},\mathsf{PKE},f}(\lambda) := \left| 2\Pr\left[\mathbf{Exp}^{lRCCA}_{\mathcal{A},\mathsf{PKE},f}(\lambda, b) = 1, b \leftarrow\$ \{0,1\}\right] - 1 \right| \in \mathsf{negl}(\lambda).$$

The ideal functionality has as parameters a public-key encryption scheme $\mathsf{PKE} := (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$, an efficiently-computable function f and (implicit) group parameters $\mathsf{pp} \in \mathsf{Setup}(1^\lambda)$. It interacts with m parties $\mathcal{P}_i$ and with an adversary S.

Public Key. Upon message $(\mathsf{KEY}, \mathsf{sid})$ from a party $\mathcal{P}_i$, $i \in [m]$, if $(\mathsf{sid}, \mathsf{pk}, \mathsf{sk})$ is not in the database sample $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\$\ \mathsf{KGen}(\mathsf{pp})$ and store the tuple $(\mathsf{sid}, \mathsf{pk}, \mathsf{sk})$ in the database. Send $(\mathsf{KEY}, \mathsf{sid}, \mathsf{pk})$ to $\mathcal{P}_i$.

Verify then Decrypt. Upon message $(\mathsf{VTDEC}, \mathsf{sid}, C_V, C_D)$ from party $\mathcal{P}_i$:

- If the tuple $(\mathsf{sid}, \mathsf{pk}, \mathsf{sk})$ does not exist in the database, ignore the message.
- Check that a tuple $(\mathsf{sid}, C_V, C_D, \mathcal{I})$ where $\mathcal{I} \subseteq [m]$ exists in the database; if so, update $\mathcal{I}$ including the index i, otherwise create the new entry $(\mathsf{sid}, C_V, C_D, \{i\})$ in the database.

If $|\mathcal{I}| = m$ and $C_D \subseteq C_V$ then:

- Send $(\mathsf{sid}, f(\mathsf{sk}))$ to the adversary S.
- Parse C_V as $(\mathsf{C}_i^V)_{i \in [|C_V|]}$ and C_D as $(\mathsf{C}_i^D)_{i \in [|C_D|]}$
- Compute the vector $\vec{b} \in \{0,1\}^{|C_V|}$ such that for any i, $b_i = 1$ if and only if $\mathsf{Dec}(\mathsf{sk}, \mathsf{C}_i^V) \neq \perp$.
- If $\exists i : b_i = 0$ set $M_o := ()$, else compute $M_o := (\mathsf{Dec}(\mathsf{sk}, \mathsf{C}_i^D))_{i \in [|C_D|]}$, send a public delayed output $(\mathsf{VTDEC}, \mathsf{sid}, \vec{b}, M_o)$ to the parties $\mathcal{P}_i$ for $i \in [m]$,

Figure 6.3: The UC ideal functionality $\mathcal{F}_{\mathsf{VtDec}}^{\mathsf{PKE},f}$ for Verify-then-Decrypt.

6.4 The Verify-then-Decrypt Ideal Functionality

We give in Fig. 6.3 the formal definition of this ideal functionality. Informally, the ideal functionality accepts two lists of ciphertexts, such that the first list includes all the ciphertexts in the second list, it first verifies that all the ciphertexts in the first list decrypt to valid messages (i.e. no decryption error) and releases such output together with the decryption from the second list. The functionality has parameter f that denotes the leakage of secret information allowed to realize such functionality.

6.5 Mix-Net

We now describe our mixnet protocol that UC-realizes the ideal functionality $\mathcal{F}_{\text{Mix}}$ with setup assumptions $\mathcal{F}_{\mathsf{VtDec}}$ and $\mathcal{F}_{\mathsf{crs}}$. We start by giving the definition of Sumcheck-Admissible relation with respect to a PKE. In this definition we abstract the necessary property for the zero-knowledge proof system used by the mixers in the protocol.

Definition 6.5.1 (Sumcheck-Admissible Relation w.r.t. PKE)**.** *Let* PKE *be a public-key encryption scheme with public space* $\mathcal{PK}$ *and the ciphertext space being a subset of* $\mathcal{CT}$*. For any* λ*, any* $\mathsf{pp} \in \mathsf{Setup}(1^\lambda)$*, let* $\mathcal{R}_{ck}^{\mathsf{pp}} : (\mathcal{PK} \times \mathcal{CT}^{2n}) \times \{0,1\}^*$ *be an NP-relation. We parse an instance of*

The functionality has n sender parties $\mathcal{P}_{S_i}$, m mixer parties $\mathcal{P}_{M_i}$.

Input. Upon activation on message $(\mathsf{INPUT}, \mathsf{sid}, \mathtt{msg})$ from $\mathcal{P}_{S_i}$ (or the adversary if $\mathcal{P}_{S_i}$ is corrupted), if $i \in L_{S,\mathsf{sid}}$ ignore the message else register the index i in the list of the senders $L_{S,\mathsf{sid}}$ and register the message $\mathtt{msg}$ in the list $L_{I,\mathsf{sid}}$ of the input messages. Notify the adversary that the sender $\mathcal{P}_{S_i}$ has sent an input.

Mix. Upon activation on message $(\mathsf{MIX}, \mathsf{sid})$ from $\mathcal{P}_{M_i}$ (or the adversary if $\mathcal{P}_{M_i}$ is corrupted), register the index i in the list of the mixers $L_{\mathrm{mix},\mathsf{sid}}$ and notify the adversary.

Delivery. Upon activation on message $(\mathsf{DELIVER}, \mathsf{sid})$ from the adversary $\mathcal{S}$ If $|L_{\mathrm{mix},\mathsf{sid}}| = m$ and $|L_{S,\mathsf{sid}}| = n$ then send a public delayed output $M_{\mathsf{sid}} \leftarrow \mathsf{Sort}(L_{I,\mathsf{sid}})$ to all the mixer parties.

Figure 6.4: The UC ideal functionality $\mathcal{F}_{\mathrm{Mix}}$ for MixNet.

The functionality interacts with n parties $\mathcal{P}_i$ and an adversary S and has parameters a PPT algorithm Init that outputs obliviously sampleable common-reference string and an (implicit) public parameter pp.

Initialization. Upon activation, sample $\mathsf{crs} \leftarrow\!\!\$\ \mathsf{Init}(\mathsf{pp})$ and store it.

Public Value. Upon activation on message CRS from a party $\mathcal{P}_i$, $i \in [n]$, send crs to $\mathcal{P}_i$.

Figure 6.5: The UC ideal functionality $\mathcal{F}_{\mathrm{CRS}}^{\mathsf{Init}}$ for Common Reference String parametrized by group parameters pp and a NIZK setup Init.

$\mathcal{R}_{ck}^{\mathsf{pp}}$ *as* $x = (\mathsf{pk}, L_1, L_2)$ *where* $L_j = (\mathtt{C}_i^j)_{i\in[n]}$ *for* $j \in \{1,2\}$. $\mathcal{R}_{ck}$ *is Sumcheck-Admissible w.r.t.* PKE *if:*

(Sumcheck) *For any* $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{pp})$ *and for any* $x := (\mathsf{pk}, L_1, L_2)$ *we have that if* $x \in \mathcal{L}(\mathcal{R}_{ck})$ *and* $\forall j, i : \mathsf{Dec}(\mathsf{sk}, \mathtt{C}_i^j) \neq \perp$ *then* $\sum_i \mathsf{Dec}(\mathsf{sk}, \mathtt{C}_i^1) - \mathsf{Dec}(\mathsf{sk}, \mathtt{C}_i^2) = 0$.

(Re-Randomization Witness) *For any* $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{pp})$ *and for any* $x := (\mathsf{pk}, L_1, L_2)$ *such that there exists* $(r_i)_{i\in[n]}$ *where* $\forall i \in [n], \exists j \in [n] : \mathtt{C}_i^2 = \mathsf{Rand}(\mathsf{pk}, \mathtt{C}_j^1; r_i)$ *we have that* $(x, (r_i)_{i\in[n]}) \in \mathcal{R}_{ck}$.

Building Blocks. Let PKE be a Rand-PKE scheme, let f be any efficiently-computable function and let $\mathcal{R}_{\mathrm{ck}}$ be any Sumcheck-Admissible relation w.r.t. PKE. The building blocks for our Mix-Net are:

1. A Rand-PKE scheme PKE that is lRCCA-secure w.r.t. f according to Definition 6.3.1.
2. An All-but-One Perfect-Sound tag-based NIZK (cf. Section 6.2) $\mathsf{NIZK}_{\mathsf{mx}} := (\mathsf{Init}_{\mathsf{mx}}, \mathsf{P}_{\mathsf{mx}}, \mathsf{V}_{\mathsf{mx}})$ for proving membership in the relation $\mathcal{R}_{\mathrm{ck}}$, with tag space $[m]$.
3. An All-but-One Perfect-Hiding tag-based NIZK $\mathsf{NIZK}_{\mathsf{sd}} = (\mathsf{Init}_{\mathsf{sd}}, \mathsf{P}_{\mathsf{sd}}, \mathsf{V}_{\mathsf{sd}})$ for knowledge of the plaintext, i.e. a NIZK for the relation $\mathcal{R}_{\mathrm{msg}} := \{(\mathsf{pk}, \mathtt{C}), (\mathtt{msg}, r) : \mathtt{C} = \mathsf{Enc}(\mathsf{pk}, \mathtt{msg}; r)\}$,

Input. Upon activation on message $(\mathsf{INPUT}, \mathsf{sid}, \mathtt{msg})$, $\mathcal{P}_{S_i}$ computes $\mathtt{C} \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{pk}, \mathtt{msg})$, and $\pi_{\mathsf{sd}} \leftarrow\!\!\$\ \mathsf{P}_{\mathsf{sd}}(\mathsf{crs}_{\mathsf{sd}}, i, (\mathsf{pk}, \mathtt{C}), (\mathtt{msg}, r))$. Broadcasts $(\mathsf{sid}, i, \mathtt{C}, \pi_{\mathsf{sd}})$.

Mix. Upon activation, the party $\mathcal{P}_{M_i}$, depending on its state, does as follow:

- If it is the first activation with message $(\mathsf{MIX}, \mathsf{sid})$ from the environment sends the message $(\mathsf{KEY}, \mathsf{sid})$ to $\mathcal{F}_{\mathsf{VtDec}}$ and return.
- If the message $(\mathsf{KEY}, \mathsf{sid}, \mathsf{pk})$, the messages $(\mathsf{sid}, i, \mathtt{C}, \pi_{\mathsf{sd}})$ for all the senders and the messages $(\mathsf{sid}, L_j, \pi^j_{\mathsf{mx}})$ for all the mixers with index $j \leq i-1$ were received:
  1. Samples a permutation ζ_i
  2. Reads the pair message $(L_{i-1}, \pi^i_{\mathsf{mx}})$ sent by the party $\mathcal{P}_{M_{i-1}}$ (or simply reads L_0 if this is the first mixer party) and parses the list L_{i-1} as $(\mathtt{C}_{i-1,j})_{j\in[n]}$.
  3. Shuffles and re-randomizes the list of ciphertexts: produces the new list $L_i \leftarrow (\mathtt{C}'_{\zeta_i(j)})_{j\in[n]}$, where $\mathtt{C}'_j \leftarrow \mathsf{Rand}(\mathsf{pk}, \mathtt{C}_{i-1,j}; r_j)$ and r_j is a uniformly random string.
  4. Computes the sumcheck proof $\pi^i_{\mathsf{mx}} \leftarrow\!\!\$\ \mathsf{P}_{\mathsf{mx}}(\mathsf{crs}_{\mathsf{mx}}, (\mathsf{pk}, L_{i-1}, L_i), (r_j)_{j\in[n]})$ for the two lists of ciphertexts.
  5. Sends to all the mixers $(\mathsf{sid}, L_i, \pi^i_{\mathsf{mx}})$.
- If the message $(\mathsf{sid}, L_m, \pi^m_{\mathsf{mx}})$ was received, checks that all the mixer proofs π^i_{mx}, for $i \in [m]$ accept, else abort.
- Computes $L \coloneqq \mathsf{Concat}(L_1, \ldots, L_m)$ and sends $(\mathsf{VtDEC}, \mathsf{sid}, L, L_m)$ to $\mathcal{F}_{\mathsf{VtDec}}$
- If the message $(\mathsf{sid}, \vec{b}, M_o)$ from $\mathcal{F}_{\mathsf{VtDec}}$ was received, if $\exists i : b_i = 0$ then returns $\perp$, else computes and returns $L_o \coloneqq \mathsf{Sort}(M_o)$

Figure 6.6: Our protocol Π_{Mix}.

with tag space $[n]$. In particular, a weaker notion of extractability that guarantees that the message $\mathtt{msg}$ is extracted is sufficient.

4. An ideal functionality $\mathcal{F}^{\mathsf{PKE},f}_{\mathsf{VtDec}}$, as defined in Fig. 6.3.
5. An ideal functionality for the common reference string (see Fig. 6.5) of the above NIZKs. In particular, the functionality initializes a CRS $\mathsf{crs}_{\mathsf{mx}}$ for $\mathsf{NIZK}_{\mathsf{mx}}$, and an additional CRS $\mathsf{crs}_{\mathsf{sd}}$ for $\mathsf{NIZK}_{\mathsf{sd}}$.

Finally, we implicitly assume that all parties have access to point-to-point authenticated channels.

Protocol Description. To simplify the exposition, we describe in this section the case of a single invocation, i.e. the protocol is run only once with a single, fixed session identifier sid; in Fig. 6.6 we describe in detail the protocol for the general case of a multi-session execution. At the first activation of the protocol, both the mixer parties and the sender parties receive from the functionality $\mathcal{F}_{\mathsf{VtDec}}$ the public key pk for the scheme PKE and the CRSs from $\mathcal{F}_{\mathrm{CRS}}$.

At submission phase, each sender $\mathcal{P}_{S_i}$ encrypts their input message $\mathtt{msg}_i$ by computing $\mathtt{C}_i \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{pk}, \mathtt{msg}_i)$, and attaches a NIZK proof of knowledge π^i_{sd} of the plaintext, using i as tag. Finally, the party $\mathcal{P}_{S_i}$ broadcasts their message $(\mathtt{C}_i, \pi^i_{\mathsf{sd}})$. After all sender parties have produced their ciphertexts, the mixers, one by one, shuffle their input lists and forward to the next mixer their output lists. In particular, the party $\mathcal{P}_{M_i}$ produces a random permutation of the input list of ciphertexts L_{i-1} (L_0 is the list of ciphertexts from the senders) by re-randomizing each ciphertext in the list and then permuting the whole list, thus computing a new list L_i. Additionally, the mixer computes a NIZK proof of membership π^i_{mx} with tag i, for the instance $(\mathsf{pk}, L_{i-1}, L_i)$ being in the sumcheck-admissible relation, because of the re-randomization witness property of Definition 6.5.1, the mixer holds a valid witness for such an instance. After this phase, the mixers are ready for the verification: the mixers invoke the Verify-then-Decrypt functionality $\mathcal{F}_{\mathsf{VtDec}}$ to (i) verify that each list seen so far is made up only of valid ciphertexts and (ii) decrypt the ciphertexts contained in the final list. Finally publishes the list of the messages received by $\mathcal{F}_{\mathsf{VtDec}}$, sorted according to some common deterministic criterion, e.g. the lexicographical order.

Theorem 6.5.1. *For any arbitrary leakage function f, if* PKE *is lRCCA-secure w.r.t. f,* $\mathsf{NIZK}_{\mathsf{mx}}$ *is ABO Perfect Sound,* $\mathsf{NIZK}_{\mathsf{sd}}$ *is ABO Perfect Hiding, then the protocol described in Fig. 6.6 UC-realizes the functionality $\mathcal{F}_{\mathrm{Mix}}$, described in Fig. 6.4, with setup assumptions $\mathcal{F}_{\mathsf{VtDec}}^{\mathsf{PKE},f}$ and $\mathcal{F}_{\mathsf{crs}}$.*

Proof. We now prove the existence of a simulator S, and we show that no PPT environment Z can distinguish an interaction with the real protocol from an interaction with S and the ideal functionality $\mathcal{F}_{\mathrm{Mix}}$ (the ideal world), i.e. the distribution $(\mathcal{F}_{\mathsf{VtDec}}, \mathcal{F}_{\mathsf{crs}})\text{-}\mathsf{Hybrid}_{Z,\Pi_{\mathrm{Mix}},\mathcal{A}}(\lambda)$ is indistinguishable from $\mathsf{Ideal}_{Z,\mathcal{F}_{\mathrm{Mix}},\mathsf{S}}(\lambda)$. In our proof, we give a sequence of hybrid experiments in which the $(\mathcal{F}_{\mathsf{VtDec}}, \mathcal{F}_{\mathsf{crs}})$-hybrid world is progressively modified until reaching an experiment that is identically distributed to the ideal world. In what follows, we indicate with h^* the index of the first honest mixer. For label $\in \{\mathrm{in}, \mathrm{hide}\}$, we introduce the set Ψ_{label} consisting of tuples (x, y). We define the functions ψ_{label} and $\psi^{-1}_{\mathrm{label}}$ associated with the corresponding set:

$$\psi_{\mathrm{label}}(x) \coloneqq \begin{cases} y & \text{if } (x,y) \in \Psi_{\mathrm{label}} \\ x & \text{otherwise} \end{cases} \qquad \psi^{-1}_{\mathrm{label}}(y) \coloneqq \begin{cases} x & \text{if } (x,y) \in \Psi_{\mathrm{label}} \\ y & \text{otherwise} \end{cases}$$

Informally, the pair of functions $\psi_{\mathrm{in}}, \psi^{-1}_{\mathrm{in}}$ helps the hybrids to keep track of the ciphertexts sent by the honest senders while they are mixed by the first $h^* - 1$ mixers, while the pair of functions $\psi_{\mathrm{hide}}, \psi^{-1}_{\mathrm{hide}}$ helps to keep track of the ciphertexts output by the first honest mixer while they are mixed by the remaining mixers in the chain. We recall that in the protocol the mixers $\mathcal{P}_{M_i}$, for $i \in [m]$, send a message which includes a list L_i of ciphertexts. Whenever it is convenient we parse L_i as $(\mathtt{C}_{i,j})_{j\in[n]}$. Let `Invalid` be the event that, during the interaction of Z with the simulator/protocol, there exist $i \in [m], j \in [n]$ such that $\mathsf{Dec}(\mathsf{sk}, \mathtt{C}_{i,j}) = \perp$ or $\mathsf{Verify}(\mathsf{crs}_{\mathsf{mx}}, (\mathsf{pk}, L_{i-1}, L_i), \pi^i_{\mathsf{mx}}) = 0$ (namely, π^i_{mx} does not verify). Clearly, when the event `Invalid` occurs, the protocol aborts.

Hybrid $\mathbf{H}_0$. This first hybrid is equivalent to $(\mathcal{F}_{\mathsf{VtDec}}, \mathcal{F}_{\mathsf{crs}})\text{-}\mathsf{Hybrid}_{Z,\Pi_{\mathrm{Mix}},\mathcal{A}}(\lambda)$.

Hybrid $\mathbf{H}_1$. In this hybrid, we change the way $\mathsf{crs}_{\mathsf{mx}}$ is generated. We run $(\mathsf{crs}_{\mathsf{mx}}, \mathsf{tps}) \leftarrow\!\!\$\ \mathsf{ABOInit}(\mathsf{pp}, h^*)$. Also, the proof $\pi^{h^*}_{\mathsf{mx}}$ of the first honest mixer is simulated. This hybrid is indistinguishable from the previous one because of the ABO Composable Perfect Zero-Knowledge property of the NIZK (cf. Section 6.2).

Hybrid $\mathbf{H}_2$. The first honest mixer $\mathcal{P}_{M_{h^*}}$, rather than re-randomizing the ciphertexts received in input, decrypts and re-encrypts all the ciphertexts. If the decryption fails for some ciphertext $\mathtt{C}_i$, $\mathcal{P}_{M_{h^*}}$ re-randomizes this "invalid" ciphertext and continues. This hybrid is indistinguishable from the previous one because the PKE scheme PKE is perfectly re-randomizable (cf. Definition 2.4.1): because of the tightness of the decryption property, we have that $\forall j$, if $\mathsf{Dec}(\mathsf{sk}, \mathtt{C}_{h^*-1,j}) = \mathtt{msg}_{h^*-1,j} \neq \perp$ then $\mathtt{C}_{h^*,j} \in \mathsf{Enc}(\mathsf{pk}, \mathtt{msg}_{h^*-1,j})$ with overwhelming probability; also, by the indistinguishability property, the distribution of the re-randomized ciphertext $\mathsf{Rand}(\mathsf{pk}, \mathtt{C}_{h^*-1,j})$ and a fresh encryption $\mathsf{Enc}(\mathsf{pk}, \mathtt{msg}_{h^*-1,j})$ are statistically close.

Hybrid $\mathbf{H}_3$. Here we introduce the set Ψ_{hide}, and we populate it with the pairs $(\mathtt{msg}_{h^*-1,i}, \mathtt{H}_i)_{i\in[n]}$, where the messages $\mathtt{H}_1, \ldots, \mathtt{H}_n$ are distinct and sampled at random from the message space $\mathcal{M}$. When we simulate the ideal functionality $\mathcal{F}_{\mathsf{VtDec}}$, we output $\psi^{-1}_{\text{hide}}(\mathtt{msg})$ for all successfully decrypted messages $\mathtt{msg}$. The only event that can distinguish the two hybrids is the event that $\neg\mathtt{Invalid}$ and $\exists j, j' : \mathsf{Dec}(\mathsf{sk}, \mathtt{C}_{m,j}) = \mathtt{H}_{j'}$. However, the messages $\mathtt{H}_1, \ldots, \mathtt{H}_n$ are not in the view of Z, thus the probability of such event is at most $\frac{n^2}{|\mathcal{M}|}$. This hybrid and the previous one are statistically indistinguishable.

Hybrid $\mathbf{H}_4$. In this hybrid, rather than re-encrypting the same messages, the first honest mixer re-encrypts the fresh and uncorrelated messages $\mathtt{H}_1, \ldots, \mathtt{H}_n$ (used to populate Ψ_{hide}). Specifically, $\mathcal{P}_{M_{h^*}}$ samples a random permutation ζ_{h^*} and computes the list $L_{h^*} := (\mathtt{C}_{h^*,j})_{j\in[n]}$, with $\mathtt{C}_{h^*,\zeta_{h^*}(j)} \leftarrow\!\$\ \mathsf{Enc}(\mathsf{pk}, \psi_{\text{hide}}(\mathtt{msg}_{h^*-1,j}))$. This hybrid is indistinguishable from the previous one, and the proof can be reduced to the lRCCA security of the scheme PKE.

Lemma 6.5.1. *Hybrids $\boldsymbol{H}_3$ and $\boldsymbol{H}_4$ are computationally indistinguishable.*

Proof. We use a hybrid argument. Let $\mathbf{H}_{3,i}$ be the hybrid game in which the first honest mixer computes the list $L_{h^*} := (\mathtt{C}_{h^*,j})_{j\in[n]}$ as:

$$\mathtt{C}_{h^*,\zeta_{h^*}(j)} := \begin{cases} \mathsf{Enc}(\mathsf{pk}, \psi_{\text{hide}}(\mathtt{msg}_{h^*-1,j})) & \text{if } j \leq i \\ \mathsf{Enc}(\mathsf{pk}, \mathtt{msg}_{h^*-1,j}) & \text{if } j > i \end{cases}$$

In particular, it holds that $\mathbf{H}_3 \equiv \mathbf{H}_{3,0}$ and $\mathbf{H}_4 \equiv \mathbf{H}_{3,n}$. We prove that $\forall i \in [n]$ the hybrid $\mathbf{H}_{3,i-1}$ is computationally indistinguishable from $\mathbf{H}_{3,i}$, reducing to the lRCCA-security of the scheme PKE. Consider the following adversary against the lRCCA-security experiment.

Adversary $\mathcal{B}(\mathsf{pk})$ with oracle access to $\mathsf{ODec}(\cdot)$.

- Simulate the hybrid experiment $\mathbf{H}_{3,i-1}$, in particular, when the environment instructs a corrupted mixer to send the message $(\mathsf{KEY}, \mathsf{sid})$ simulate the ideal functionality $\mathcal{F}_{\mathsf{VtDec}}$ sending back the answer $(\mathsf{KEY}, \mathsf{sid}, \mathsf{pk})$.
- When it is time to compute the list of the first honest mixer L_{h^*}, namely, when the mixer $\mathcal{P}_{M_{h^*}}$ is activated by the environment and has received for all $j \in [n]$ the messages $(\mathsf{sid}, j, \mathtt{C}, \pi_{\mathsf{sd}})$ from the senders and the messages $(\mathsf{sid}, L_j, \pi^j_{\mathsf{mx}})$ from all the mixers with index $j \leq h^* - 1$, first decrypt all the ciphertexts received so far using $\mathsf{ODec}(\cdot)$. Let $\mathtt{msg}_{h^*-1,i}$ be the decryption of the ciphertext $\mathtt{C}_{h^*-1,i}$. If $\mathtt{msg}_{h^*-1,i} = \perp$ then output a random bit, else send the pair of messages $(\mathtt{msg}_{h^*-1,i}, \mathtt{H}_i)$ to the lRCCA challenger, thus receiving a challenge ciphertext $\mathtt{C}^*$.

- Populate the list L_{h^*} by setting $\mathtt{C}_{\zeta_{h^*}(i)} \leftarrow \mathtt{C}^*$, and computing all the other ciphertexts as described in $\mathbf{H}_{3,i-1}$. Continue the simulation as the hybrid does.
- When all the mixers have sent the message (VtDEC, L, L_m), to $\mathcal{F}_{\mathsf{VtDec}}$, check that all the mixer proofs accept, otherwise abort the simulation and output a random bit. Then decrypt all the ciphertexts in L by sending queries to the guarded decryption oracle, i.e. send the query $\mathtt{C}_{i',j}$, receiving back the message $\mathtt{msg}_{i',j} \in \mathcal{M} \cup \{\diamond, \perp\}$. If $\mathtt{msg}_{i',j} = \perp$, abort and output a random bit. If $\mathtt{msg}_{i',j} = \diamond$, then set $\mathtt{msg}_{i',j} \coloneqq \mathtt{msg}_{h^*-1,i}$. Simulate the leakage from $\mathcal{F}_{\mathsf{VtDec}}$ through the leakage received by the lRCCA security experiment: in particular, the reduction loses access to the guarded decryption oracle, receives the value $f(\mathsf{sk})$ and sends the message $(\mathsf{sid}, \vec{b}, \{\mathtt{msg}_{m,j}\}_{j\in[n]})$ as required by the protocol.
- Finally, forward the bit returned by Z.

First we notice that when the guarded decryption oracle returns a message $\mathtt{msg}_{i',j} = \diamond$ then the reduction can safely return $\mathtt{msg}_{h^*-1,i}$. In fact, the ciphertext would decrypt to either $\mathtt{H}_i$ or to $\mathtt{msg}_{h^*-1,i}$, however by the change introduced in $\mathbf{H}_3$, we have that $\mathtt{msg}_{h^*-1,i} = \psi^{-1}_{\text{hide}}(\mathtt{H}_i)$ and $\mathtt{msg}_{h^*-1,i} = \psi^{-1}_{\text{hide}}(\mathtt{msg}_{h^*-1,i})$.

It is easy to see that when the challenge bit b of the experiment is equal to 0, the view of Z is identically distributed to the view in $\mathbf{H}_{3,j-1}$, while if the challenge bit is 1, the view of Z is identically distributed to the one in $\mathbf{H}_{3,j}$. Thus, $|\Pr[\mathbf{H}_{3,j-1}(\lambda) = 1] - \Pr[\mathbf{H}_{3,j}(\lambda) = 1]| \leq \mathbf{Adv}^{lRCCA}_{\mathcal{B},\mathsf{PKE},f}(\lambda)$. □

Hybrid $\mathbf{H}_5$. Let $V_m \coloneqq (\mathtt{msg}_{m,j})_{j\in[n]}$ (resp. $V_{h^*} \coloneqq (\mathtt{msg}_{h^*,j})_{j\in[n]}$) be the list of decrypted ciphertexts output by the last mixer $\mathcal{P}_{M_m}$ (resp. by the first honest mixer $\mathcal{P}_{M_{h^*}}$). In the hybrid $\mathbf{H}_5$ the simulation aborts if $\neg\mathtt{Invalid}$ and $V_m \neq V_{h^*}$.

Lemma 6.5.2. *Hybrids $\boldsymbol{H}_4$ and $\boldsymbol{H}_5$ are computationally indistinguishable.*

Proof. Since $|V_m| = |V_{h^*}|$ and the messages $\mathtt{H}_1, \ldots, \mathtt{H}_n$ are distinct, the event $V_{h^*} \neq V_m$ holds if and only if there exists an index $j \in [n]$ such that $\mathsf{Count}(\mathtt{H}_j, V_m) \neq 1$. Let $\mathbf{H}_{4,i}$ be the same as $\mathbf{H}_4$ but the simulation aborts if $\neg\mathtt{Invalid}$ and $\exists j \in [i] : \mathsf{Count}(\mathtt{H}_j, V_m) \neq 1$. Clearly, $\mathbf{H}_{4,0} \equiv \mathbf{H}_4$ and $\mathbf{H}_{4,n} \equiv \mathbf{H}_5$. Let $\mathtt{Bad}_i$ be the event that $(\neg\mathtt{Invalid} \wedge \mathsf{Count}(\mathtt{H}_i, V_m) \neq 1)$. It is easy to check that:

$$|\Pr[\mathbf{H}_{4,i-1}(\lambda) = 1] - \Pr[\mathbf{H}_{4,i}(\lambda) = 1]| \leq \Pr[\mathtt{Bad}_i].$$

In fact, the two hybrids are equivalent if the event $\mathtt{Bad}_i$ does not happen.

We define an adversary to the lRCCA security of PKE that makes use of the event above.

Adversary $\mathcal{B}(\mathsf{pk})$ with oracle access to $\mathsf{ODec}(\cdot)$.

1. Simulate the hybrid experiment $\mathbf{H}_5$; in particular, when the environment instructs a corrupted mixer to send the message $(\mathsf{KEY}, \mathsf{sid})$ simulate the ideal functionality $\mathcal{F}_{\mathsf{VtDec}}$ sending back the answer $(\mathsf{KEY}, \mathsf{sid}, \mathsf{pk})$. (Thus embedding the public key from the challenger in the simulation.)
2. When it is time to compute the list of the first honest mixer L_{h^*}, namely, when the mixer $\mathcal{P}_{M_{h^*}}$ is activated by the environment and has received the messages $(\mathsf{sid}, i, \mathtt{C}, \pi_{\mathsf{sd}})$ for all the senders and the messages $(\mathsf{sid}, L_j, \pi^j_{\mathsf{mx}})$ for all

the mixers with index $j \leq h^* - 1$, first decrypt all the ciphertexts received so far using the guarded decryption oracle. If there is a decryption error, output a random bit b'.

3. Sample $\mathtt{H}^{(0)}, \mathtt{H}^{(1)} \leftarrow\!\!\$\ \mathcal{M}$ and send the pair of messages $(\mathtt{H}^{(0)}, \mathtt{H}^{(1)})$ to the lR-CCA challenger, receiving back the challenge ciphertext $\mathtt{C}^*$. Set the list $L_{h^*} = (\mathtt{C}_{h^*,j})_{j\in[n]}$ as follows:

$$\mathtt{C}_{h^*,\zeta_{h^*}(j)} := \left\{ \begin{array}{cc} \mathsf{Enc}(\mathsf{pk}, \mathtt{msg}_{h^*-1,j}) & \text{if } j \neq i \\ \mathtt{C}^* & \text{else} \end{array} \right\}$$

where recall that ζ_{h^*} is the random permutation used by the h^*-th mixer. Continue the simulation as the hybrid does.

4. When all the mixer have sent the message (VtDEC, L, L_m), to $\mathcal{F}_{\mathsf{VtDec}}$, decrypt all the ciphertexts in L by sending queries to the guarded decryption oracle, namely, send the query $\mathtt{C}_{i',j}$ for all $i' > h^*$ and all $j \in [n]$, receiving back as answer the plaintext messages $\mathtt{msg}_{i',j} \in \mathcal{M} \cup \{\diamond, \perp\}$.
If the event $\mathtt{Invalid}$ holds, then abort the simulation and output a random bit b'.

5. Let $C \leftarrow \mathsf{Count}(\diamond, V_m)$, if $C = 1$ then abort the simulation and output a random bit b'.

6. From now one we can assume that $\neg\mathtt{Invalid}$ and $C \neq 1$; Compute

$$\mathtt{msg} \leftarrow (C-1)^{-1} \cdot \left(\sum_{j\in[n], \mathtt{msg}_{m,j} \neq \diamond} \mathtt{msg}_{m,j} - \sum_{j \neq \zeta_{h^*}(i)} \mathtt{msg}_{h^*,j} \right). \tag{6.1}$$

Output b' s.t. $\mathtt{msg} = \mathtt{H}^{(b')}$.

First, we notice that the simulation $\mathcal{B}$ provides to the environment Z is perfect, indeed, independently of the challenge bit, the message $\mathtt{H}^{(b)}$ is distributed identically to $\mathtt{H}_j$. Thus, the probability that $\mathtt{Bad}_i$ happens in the reduction is the same as the probability the event happens in the hybrid experiments.

Let $\mathtt{Abort}$ be the event that $\mathcal{B}$ aborts and outputs a random bit. Notice that:

$$\mathtt{Abort} \equiv \mathtt{Invalid} \vee (C = 1).$$

Let $\mathtt{Wrong}$ be the event that $\exists j : \mathsf{Dec}(\mathsf{sk}, \mathtt{C}_{m,j}) = \mathtt{H}^{(1-b)}$; notice that the message $\mathtt{H}^{(1-b)}$ is independent of the view of the environment Z, thus the probability of $\mathtt{Wrong}$ is at most $n/|\mathcal{M}|$. Moreover, we have $\mathtt{Bad}_i \equiv \neg\mathtt{Abort} \wedge \neg\mathtt{Wrong}$ because, by definition of $\neg\mathtt{Wrong}$, all the ciphertexts that decrypt to $\diamond$ in L_m are indeed an encryption of $\mathtt{H}^{(b)}$; thus, assuming the event holds, $C \neq 1$ if and only if $\mathsf{Count}(\mathtt{H}^{(b)}, V_m) \neq 1$. The probability of guessing the challenge bit when $\mathcal{B}$ aborts is $\frac{1}{2}$, thus we have:

$$\Pr[b = b'] \geq \tfrac{1}{2} \Pr[\neg\mathtt{Bad}_i] + \Pr[b = b' | \mathtt{Bad}_i] \Pr[\mathtt{Bad}_i] - \tfrac{n}{|\mathcal{M}|} \tag{6.2}$$

We now compute the probability that $b = b'$ conditioned on $\mathtt{Bad}_i$. First notice that $\neg\mathtt{Invalid}$ implies that the ciphertexts in the lists $L_{h^*}, \ldots, L_m$ decrypt correctly and that the proofs π^j_{mx}

for $j > h^*$ verify. Thus, by applying the sumcheck-admissibility w.r.t. PKE of the relation $\mathcal{R}_{\mathsf{mx}}$ and by the ABO perfect soundness of $\mathsf{NIZK}_{\mathsf{mx}}$ we have:

$$\sum_{j\in[n]} \mathsf{Dec}(\mathsf{sk}, \mathtt{C}_{h^*,j}) - \sum_{j\in[n]} \mathsf{Dec}(\mathsf{sk}, \mathtt{C}_{m,j}) = 0.$$

If we condition on $\neg\mathtt{Wrong}$ then:

$$\left(\mathtt{H}^{(b)} + \sum_{j\neq\zeta_{h^*}(j^*)} \mathtt{msg}_{h^*,j}\right) - \left(C\cdot\mathtt{H}^{(b)} + \sum_{j\in[n],\mathtt{msg}_{m,j}\neq\diamond} \mathtt{msg}_{m,j}\right) = 0.$$

By solving the above equation for $\mathtt{H}^{(b)}$, we obtain $\mathtt{msg} = \mathtt{H}^{(b)}$, therefore $\mathcal{B}$ guesses the challenge bit with probability 1 when conditioning on $\neg\mathtt{Abort} \wedge \neg\mathtt{Wrong}$. □

Hybrid $\mathbf{H}_6$. In this hybrid, we modify the decryption phase. When for all $j \in [m]$ the mixer has sent $(\mathsf{VtDEC}, \mathsf{sid}, L, L_m)$ to $\mathcal{F}_{\mathsf{VtDec}}$, the hybrid simulates the answer of the ideal functionality sending the message $(\mathsf{sid}, \vec{b}, M'_o)$ where $\vec{b}$ is computed as defined by the ideal functionality $\mathcal{F}_{\mathsf{VtDec}}$ and M'_o is the empty list () if $\mathtt{Invalid}$ occurs; else, if all the messages in L correctly decrypt and the mixer proofs are valid, compute $M'_o \leftarrow (\mathtt{msg}_{h^*-1,\zeta_o(j)})_{j\in[n]}$, where ζ_o is a uniformly random permutation. Notice that $\mathbf{H}_6$ does not use the map ψ^{-1}_{hide} at decryption phase.

We show that this hybrid and the previous one are equivalently distributed. First, by the change introduced in the previous hybrid, if the hybrid does not abort then $V_m = V_{h^*-1}$. Moreover, the two sets below are equivalently distributed:

$$\{(\mathtt{msg}_{h^*-1,j}, \mathtt{H}_j) : j \in [n]\} \equiv \{(\mathtt{msg}_{h^*-1,j}, \mathtt{H}_{\zeta_o(j)} : j \in [n])\}$$

because the messages $\mathtt{H}_1, \ldots, \mathtt{H}_n$ are uniformly distributed.

Hybrid $\mathbf{H}_7$. Similarly to what done in $\mathbf{H}_3$, in this hybrid we introduce the set Ψ_{in}, and we populate it with the pairs $(\mathtt{msg}_i, \tilde{\mathtt{msg}}_i)_{i\leq[n]}$, where the messages $\mathtt{msg}_i$ are the inputs of the honest senders, and the messages $\tilde{\mathtt{msg}}_i$ are distinct and sampled uniformly at random from the message space $\mathcal{M}$. When we simulate the ideal functionality $\mathcal{F}_{\mathsf{VtDec}}$, in case all the ciphertexts decrypts, we output the list $M_o \coloneqq (\mathtt{msg}_{o,i})_i$, where $\mathtt{msg}_{o,\zeta_o(i)} \leftarrow \psi^{-1}_{\text{in}}(\mathtt{msg}_{h^*-1,i})$. We notice that if $V_{h^*-1} \cap \mathcal{M}_H \neq \emptyset$, the map ψ^{-1}_{in} would modify the returned value; however, since the messages $\tilde{\mathtt{msg}}_i$ are not in the view of Z, there is a probability of at most $\frac{n^2}{|\mathcal{M}|}$ that this event happens and that Z distinguishes $\mathbf{H}_6$ from $\mathbf{H}_7$.

Hybrid $\mathbf{H}_8$. In this hybrid, we encrypt the simulated (honest) sender inputs $\tilde{\mathtt{msg}}_j$ instead of the (honest) sender inputs $\mathtt{msg}_j$ to populate the list L_0. The proof that this hybrid and the previous one are computationally indistinguishable follows by the lRCCA security of PKE and the zero-knowledge of $\mathsf{NIZK}_{\mathsf{sd}}$.

Lemma 6.5.3. *Hybrids $\boldsymbol{H}_7$ and $\boldsymbol{H}_8$ are computationally indistinguishable.*

Proof. First, we switch to hybrid $\mathbf{H}'_7$ and $\mathbf{H}'_8$ that are exactly the same but where the $\mathsf{crs}_{\mathsf{sd}}$ is sampled with $\mathsf{ABOInit}(j)$ for an arbitrary index j for a corrupted party. The hybrids can be shown indistinguishable based on the lRCCA-security of the scheme PKE, and the zero-knowledge of $\mathsf{NIZK}_{\mathsf{sd}}$. We can use a hybrid argument. Let $\mathbf{H}'_{7,j}$ be the hybrid game in which we

encrypt the simulated sender inputs $\tilde{\mathtt{msg}}_i$, for $i \leq j$, and we encrypt the honest sender inputs $\mathtt{msg}_i$ for $i > j$. In particular, $\mathbf{H}'_7 \equiv \mathbf{H}'_{7,0}$ and $\mathbf{H}'_8 \equiv \mathbf{H}'_{7,n}$. We can prove that the hybrids $\mathbf{H}'_{7,j-1}$ is indistinguishable from $\mathbf{H}'_{7,j}, \forall j \in [n]$. In particular, when the j-th party is corrupt, the two hybrids are identically distributed. We focus on the more interesting case when the j-th sender party is honest.

Adversary $\mathcal{B}(\mathsf{pk})$ with oracle access to $\mathsf{ODec}(\cdot)$.

1. Start simulating the ideal functionality $\mathcal{F}_{\mathsf{VtDec}}$ sending the answer $(\mathsf{KEY}, \mathsf{sid}, \mathsf{pk})$ to the mixer parties, when instructed by the environment, thus embedding the public key from the challenger in the simulation.
2. When the honest sender party $\mathcal{P}_{S_i}$ is activated by Z on input $(\mathsf{INPUT}, \mathtt{msg}_i)$, if $i < j$ sample a random message $\tilde{\mathtt{msg}}_i$, encrypt $\tilde{\mathtt{msg}}_i$, and add the pair $(\mathtt{msg}_i, \tilde{\mathtt{msg}}_i)$ to the set Ψ_{in}, and finally simulate the proof π^i_{sd}. Instead, if $i > j$, behave like in $\mathbf{H}_8$ encrypting the honest sender input $\mathtt{msg}_i$. For $i = j$, sample a random message $\tilde{\mathtt{msg}}_j$ and submit the pair $(\mathtt{msg}_j, \tilde{\mathtt{msg}}_j)$ to the lRCCA challenger, thus receiving back the challenge ciphertext $\mathtt{C}^*$; produce a simulated proof π^j_{sd} to be attached to $\mathtt{C}^*$ and continue the simulation.
3. When all the mixers have sent the message (VtDEC, L, L_m) to $\mathcal{F}_{\text{VtDec}}$, decrypt all the ciphertexts in L by sending queries to the guarded decryption oracle, i.e. send the query $\mathtt{C}_{i',j}$, receiving back the message $\mathtt{msg}_{i',j} \in \mathcal{M} \cup \{\diamond, \perp\}$ and if $\mathtt{msg}_{i',j} = \diamond$ then set $\mathtt{msg}_{i',j} \coloneqq \psi_{\text{in}}^{-1}(\tilde{\mathtt{msg}}_j)$. If a decryption error is returned for any of the queries, or any of the proofs attached to the ciphertexts are not valid (i.e. in case of `Invalid`), abort and output a random bit b'. Else, simulate the leakage from $\mathcal{F}_{\mathsf{VtDec}}$ through the leakage received by the lRCCA security experiment; receive the value $f(\mathsf{sk})$ and send the message $(\mathsf{sid}, \vec{b}, (\mathtt{msg}_{h^*-1,\zeta_o(j)})_{j\in[n]})$ as described by the hybrid.
4. Finally, when the simulation is complete, outputs the same as Z.

We notice that if the challenge bit b of the lRCCA game is equal to 0, the simulation of $\mathcal{B}$ offered to Z is identically distributed to the view in $\mathbf{H}'_{7,j-1}$, while if the challenge bit is 1, Z is given a view identically distributed to the one in $\mathbf{H}_{7,j}$. Also, whenever the event `Invalid` occurs, the reduction $\mathcal{B}$ outputs a random bit, so conditioning on `Invalid` the advantage in the lRCCA security game is equal to 0. We have that for an environment Z, $\mathbf{Adv}^{lRCCA}_{\mathcal{B},\mathsf{PKE},f}(\lambda) = |\Pr\left[\mathbf{H}'_{7,j-1} = 1\right] - \Pr[\mathbf{H}_{7,j} = 1]|$. □

We now introduce the latest two hybrids that ensure that none of the inputs of the honest senders is duplicated or discarded: we start by introducing a check on malicious senders, while in $\mathbf{H}_{10}$ we ensure that no malicious mixer can duplicate or discard the honest inputs.

Hybrid $\mathbf{H}_9$. Let $\mathcal{M}_H$ be the set of simulated messages $\{\tilde{\mathtt{msg}}_i\}_{i\leq[n]}$ for the honest sender parties and let V_0 be the decryption of the list of ciphertexts received by the first mixer. If $\neg$`Invalid` and a message $\mathtt{msg} \in \mathcal{M}_H$ appears more than once in the list V_0 then the simulation aborts.

Lemma 6.5.4. *Hybrids $\boldsymbol{H}_8$ and $\boldsymbol{H}_9$ are computationally indistinguishable.*

Proof. We rely on the lRCCA security of PKE and the soundness of $\mathsf{NIZK}_{\mathsf{sd}}$, that indeed prevents the adversary from sending valid ciphertexts whose messages are correlated with the honest ones, to show that this abort happens only with negligible probability. Let $\mathbf{H}_{8,i}$ be the hybrid that aborts if $\neg\mathtt{Invalid}$ and a message $\mathtt{msg} \in \mathcal{M}_H$ appears more than once in the list $(\mathtt{msg}_{0,j})_{j\leq i}$. Clearly, $\mathbf{H}_{8,0} \equiv \mathbf{H}_8$ and $\mathbf{H}_{8,n} \equiv \mathbf{H}_9$. Next, let $\mathbf{H}'_{8,i}$ be the same as $\mathbf{H}_{8,i}$ but where the common reference string $\mathsf{crs}_{\mathsf{sd}}$ is sampled using $\mathsf{ABOInit}(i)$. We now show that $\mathbf{H}'_{8,i-1}$ is indistinguishable from $\mathbf{H}'_{8,i}$ for all $i \in [n]$. Notice that only difference between them is when one hybrid aborts while the other does not. Let $\mathtt{Bad}_i$ be the event that $(\neg\mathtt{Invalid} \wedge \mathtt{msg}_{0,i} \in \mathcal{M}_H)$. It holds that:

$$|\Pr\left[\mathbf{H}'_{8,i-1}(\lambda) = 1\right] - \Pr\left[\mathbf{H}'_{8,i}(\lambda) = 1\right]| \leq n \cdot \Pr[\mathtt{Bad}_i].$$

We focus on the case where i is the index of a malicious sender, as the other case is obvious. We can show a reduction to the lRCCA security of PKE.

Adversary $\mathcal{B}(\mathsf{pk})$ with oracle access to $\mathsf{ODec}(\cdot)$.

1. Simulate the hybrid experiment $\mathbf{H}'_{8,i}$; in particular, when Z instructs a corrupted mixer to send the message $(\mathsf{KEY}, \mathsf{sid})$, simulate $\mathcal{F}_{\mathsf{VtDec}}$ sending back the answer $(\mathsf{KEY}, \mathsf{sid}, \mathsf{pk})$, embedding the public key from the challenger in the simulation. Also, sample $\mathsf{crs}_{\mathsf{sd}} \leftarrow\!\!\$\ \mathsf{ABOInit}(i)$.
2. Sample $\tilde{\mathtt{msg}}^{(0)}, \tilde{\mathtt{msg}}^{(1)} \leftarrow\!\!\$\ \mathcal{M}$ and send the pair of messages $(\tilde{\mathtt{msg}}^{(0)}, \tilde{\mathtt{msg}}^{(1)})$ to the lRCCA challenger, receiving back the challenge ciphertext $\mathtt{C}^*$.
3. Sample an index $h \in [n]$, such that the h-th sender is honest.
4. When the honest sender party $\mathcal{P}_{S_j}$ is activated on input $(\mathsf{INPUT}, \mathsf{sid}, \mathtt{msg}_j)$, if $j \neq h$ sample a random message $\tilde{\mathtt{msg}}_j$ (and populate the set Ψ_{in}), compute $\mathtt{C} \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{pk}, \tilde{\mathtt{msg}}_j)$, the honest proof π^j_{sd} (as in $\mathbf{H}_8$) and send to the other parties $(\mathsf{sid}, j, \mathtt{C}_j, \pi^j_{\mathsf{sd}})$. For $j = h$, instead, send $(\mathsf{sid}, h, \mathtt{C}^*, \tilde{\pi}^h_{\mathsf{sd}})$, where the proof $\tilde{\pi}^h_{\mathsf{sd}}$ is simulated. Wait for all the senders to broadcast their messages $(\mathsf{sid}, j, \mathtt{C}_j, \pi^j_{\mathsf{sd}})$ and continue the simulation.
5. When all the mixers have sent the message $(\mathsf{VtDECsid}, L, L_m)$, decrypt all the ciphertexts in the list L by sending queries to the guarded decryption oracle, namely, send the query $\mathtt{C}_{k,j}$ for all $k \in [m]$ and all $j \in [n]$, receiving back as answer the plaintext messages $\mathtt{msg}_{k,j} \in \mathcal{M} \cup \{\diamond, \perp\}$.
   If the event $\mathtt{Invalid}$ holds, then abort the simulation and output a random bit b'.
6. If $\mathtt{msg}_{0,i} \neq \diamond$ then abort the simulation and output a random bit b'.
7. From now one we can assume that $\neg\mathtt{Invalid}$ and $\mathtt{msg}_{0,i} \in \{\tilde{\mathtt{msg}}^{(0)}, \tilde{\mathtt{msg}}^{(1)}\}$; extract from the proof π^i_{sd} the plaintext message $\mathtt{msg}$. Output b' s.t. $\mathtt{msg} = \tilde{\mathtt{msg}}^{(b')}$. If the extraction fails, output a random bit.

First, we notice that the simulation $\mathcal{B}$ provides to the environment Z is perfect: independently of the challenge bit b, the message $\tilde{\mathtt{msg}}^{(b)}$ is distributed identically to $\tilde{\mathtt{msg}}_h$, and the simulated proof is indistinguishable from the honest one, due to the ABO Zero-Knowledge property of $\mathsf{NIZK}_{\mathsf{sd}}$. Thus, the probability that $\mathtt{Bad}_i$ happens in the reduction is the same as

the probability the event happens in the hybrid experiments, conditioned on the fact that the guess of index h is correct (i.e. our strategy works with probability $\frac{1}{n}$).

With an analysis similar to what done in Lemma 6.5.2, we can easily prove that the event $\texttt{Wrong}$, i.e. the fact that $\texttt{msg}_{0,i} = \tilde{\texttt{msg}}^{(1-b)}$, only happens with negligible probability $\frac{1}{|\mathcal{M}|}$. Conditioning on $\neg\texttt{Wrong} \wedge \neg\texttt{Abort}$, we have that $\mathcal{B}$ always outputs the correct bit $b' = b$. □

Hybrid $\mathbf{H}_{10}$. Recall that $V_{h^*} := (\texttt{msg}_{h^*,j})_{j\in[n]}$ is the list of decrypted ciphertexts output by the first honest mixer $\mathcal{P}_{M_{h^*}}$. In the hybrid $\mathbf{H}_{10}$ the simulation aborts if $\neg\texttt{Invalid}$ and $\exists i \in [n]$ such that $\mathsf{Count}(\tilde{\texttt{msg}}_i, V_{h^*-1}) \neq 1$, i.e., some of the simulated honest inputs do not appear or appear more than once, encrypted, in the list received in input by the first honest mixer. With this check we ensure that none of the inputs of the honest senders has been discarded or duplicated by the (malicious) mixers.

Lemma 6.5.5. *Hybrids $\boldsymbol{H}_9$ and $\boldsymbol{H}_{10}$ are computationally indistinguishable.*

Proof. First, we switch to hybrid $\mathbf{H}'_9$ and $\mathbf{H}'_{10}$ that are exactly the same but where the $\mathsf{crs}_{\mathsf{sd}}$ is sampled with $\mathsf{ABOInit}(j)$ for an arbitrary index j for a corrupted party.

We prove this using a hybrid argument. Let $\mathbf{H}_{9',i}$ the hybrid in which the simulation aborts if $\neg\texttt{invalid}$ and $\exists j \leq i$ such that $\mathsf{Count}(\tilde{\texttt{msg}}_j, V_{h^*-1}) \neq 1$. Clearly we have that $\mathbf{H}'_9 \equiv \mathbf{H}'_{9,0}$ and $\mathbf{H}'_{10} \equiv \mathbf{H}'_{9,n}$. We now show that for all $i \in [n]$, the hybrid $\mathbf{H}'_{9,i-1}$ is indistinguishable from $\mathbf{H}'_{9,i}$. This is trivially true when the i-th sender is corrupted ($\mathbf{H}'_{9,i-1}$ and $\mathbf{H}'_{9,i}$ in this case are identically distributed).

Let $\texttt{Bad}_i$ be the event that $(\neg\texttt{Invalid} \wedge \mathsf{Count}(\tilde{\texttt{msg}}_i, V_{h^*-1}) = 0)$. It is easy to check that:

$$|\Pr\left[\mathbf{H}'_{9,i-1}(\lambda) = 1\right] - \Pr\left[\mathbf{H}'_{9,i}(\lambda) = 1\right]| \leq \Pr[\texttt{Bad}_i].$$

In fact, the two hybrids are equivalent if the event $\texttt{Bad}_i$ does not happen.

We define an adversary to the lRCCA security of PKE that makes use of the event above.

Adversary $\mathcal{B}(\mathsf{pk})$ with oracle access to $\mathsf{ODec}(\cdot)$.

1. Simulate the hybrid experiment $\mathbf{H}'_{9,i}$; in particular, when the environment instructs a corrupted mixer to send the message $(\mathsf{KEY}, \mathsf{sid})$, simulate the ideal functionality $\mathcal{F}_{\mathsf{VtDec}}$ sending back the answer $(\mathsf{KEY}, \mathsf{sid}, \mathsf{pk})$, embedding the public key from the challenger in the simulation.
2. Sample $\tilde{\texttt{msg}}^{(0)}, \tilde{\texttt{msg}}^{(1)} \leftarrow\!\!\$\ \mathcal{M}$ and send the pair of messages $(\tilde{\texttt{msg}}^{(0)}, \tilde{\texttt{msg}}^{(1)})$ to the lRCCA challenger, receiving back the challenge ciphertext $\texttt{C}^*$.
3. When the honest sender party $\mathcal{P}_{S_j}$ is activated on input $(\mathsf{INPUT}, \mathsf{sid}, \texttt{msg}_j)$, if $i \neq j$ sample a random message $\tilde{\texttt{msg}}_j$ (and populate the set Ψ_{in}), encrypt the message $\tilde{\texttt{msg}}_j$ as in $\mathbf{H}_{10}$ and continue the simulation by sending to the other parties $(\mathsf{sid}, j, \texttt{C}_j, \pi^j_{\mathsf{sd}})$. For $j = i$, instead, send $(\mathsf{sid}, i, \texttt{C}^*, \tilde{\pi}^j_{\mathsf{sd}})$, where the proof $\tilde{\pi}^j_{\mathsf{sd}}$ is simulated. Wait for all the senders to broadcast their messages $(\mathsf{sid}, j, \texttt{C}_j, \pi^j_{\mathsf{sd}})$ and continue the simulation.
4. When all the mixers have sent their message (sid, L_j), decrypt all the ciphertexts in those lists by sending queries to the guarded decryption oracle, namely, send the query $\texttt{C}_{i',j}$ for all $i' \in [m]$ and all $j \in [n]$, receiving back as answer the plaintext messages $\texttt{msg}_{i',j} \in \mathcal{M} \cup \{\diamond, \perp\}$.

If the event $\texttt{Invalid}$ holds, then abort the simulation and output a random bit b'.

5. Let $C \leftarrow \mathsf{Count}(\diamond, V_{h^*-1})$, if $C = 1$ then abort the simulation and output a random bit b'.
6. From now one we can assume that $\neg\texttt{Invalid}$ and $C \neq 1$;Compute

$$\texttt{msg} \leftarrow (C-1)^{-1} \cdot \left(\sum_{j \in [n], \texttt{msg}_{0,j} \neq \diamond} \texttt{msg}_{0,j} - \sum_{j \neq \zeta_{h^*-1}(i)} \texttt{msg}_{h^*-1,j} \right). \tag{6.3}$$

Output b' s.t. $\texttt{msg} = \tilde{\texttt{msg}}^{(b')}$.

First, we notice that the simulation $\mathcal{B}$ provides to the environment Z is perfect, indeed, independently of the challenge bit, the message $\tilde{\texttt{msg}}^{(b)}$ is distributed identically to $\tilde{\texttt{msg}}_j$. Thus, the probability that $\texttt{Bad}_i$ happens in the reduction is the same as the probability the event happens in the hybrid experiments.

Let $\texttt{Abort}$ be the event that $\mathcal{B}$ aborts and outputs a random bit. Notice that:

$$\texttt{Abort} \equiv \texttt{Invalid} \vee (C = 1).$$

Let $\texttt{Wrong}$ be the event that $\exists j : \mathsf{Dec}(\mathsf{sk}, \mathtt{C}_{h^*-1,j}) = \tilde{\texttt{msg}}^{(1-b)}$, notice that the message $\tilde{\texttt{msg}}^{(1-b)}$ is independent of the view of the environment Z, thus the probability of $\texttt{Wrong}$ is at most $n/|\mathcal{M}|$. Moreover, we have $\texttt{Bad}_i \equiv \neg\texttt{Abort} \wedge \neg\texttt{Wrong}$ because, by definition of $\neg\texttt{Wrong}$, all the ciphertexts that decrypt to $\diamond$ in L_{h^*-1} are indeed an encryption of $\tilde{\texttt{msg}}^{(b)}$, thus assuming the event holds then $C \neq 1$ if and only if $\mathsf{Count}(\tilde{\texttt{msg}}^{(b)}, V_{h^*-1}) \neq 1$. The probability of guessing the challenge bit when $\mathcal{B}$ aborts is $\frac{1}{2}$, thus we have:

$$\Pr[b = b'] \geq \tfrac{1}{2} \Pr[\neg\texttt{Bad}_i] + \Pr[b = b' | \texttt{Bad}_i] \Pr[\texttt{Bad}_i] - \tfrac{n}{|\mathcal{M}|} \tag{6.4}$$

We now compute the probability that $b = b'$ conditioned on the event $\texttt{Bad}_i$. First, we notice that $\neg\texttt{Invalid}$ implies that the ciphertexts in the lists $L_0, \ldots, L_{h^*-1}$ decrypt correctly and that the proofs π^j_{mx} for $j < h^*$ verify. Thus, by applying the sumcheck-admissibility w.r.t. PKE of the relation $\mathcal{R}_{\mathsf{mx}}$ and by the ABO perfect soundness of $\mathsf{NIZK}_{\mathsf{mx}}$ we have:

$$\sum_{j \in [n]} \mathsf{Dec}(\mathsf{sk}, \mathtt{C}_{0,j}) - \sum_{j \in [n]} \mathsf{Dec}(\mathsf{sk}, \mathtt{C}_{h^*-1,j}) = 0.$$

If we condition on $\neg\texttt{Wrong}$ then:

$$\left(\tilde{\texttt{msg}}^{(b)} + \sum_{j \neq \zeta_0(j^*)} \texttt{msg}_{0,j} \right) - \left(C \cdot \tilde{\texttt{msg}}^{(b)} + \sum_{j \in [n], \texttt{msg}_{h^*-1,j} \neq \diamond} \texttt{msg}_{h^*-1,j} \right) = 0.$$

By solving the above equation for $\tilde{\texttt{msg}}^{(b)}$, we obtain that $\texttt{msg} = \tilde{\texttt{msg}}^{(b)}$. We recall that the index $\tau^* \in [n]$ is (only) computationally hidden: $\mathcal{B}$ is able to extract from the proof $\pi^{i'}_{\mathsf{sd}}$ with probability $\frac{1}{n} - \mathsf{negl}(\lambda)$. Therefore, $\mathcal{B}$ guesses the challenge bit with probability 1 when conditioning on $\neg\texttt{Abort} \wedge \neg\texttt{Wrong} \wedge \tau^* = i'$. □

Simulator S.

Initialization. Simulate the ideal functionality $\mathcal{F}_{\mathsf{crs}}$ by sampling $\mathsf{crs}_{\mathsf{mx}}$ in ABO Perfect Sound mode on the tag h^*, while $\mathsf{crs}_{\mathsf{sd}}$ is honestly generated with $\mathsf{Init}(1^\lambda)$. Also, simulate $\mathcal{F}_{\mathsf{VtDec}}$ by a sampling key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{pp})$. Populate the set $\mathcal{M}_H$ of the simulated honest inputs, by sampling uniformly random (and distinct) messages from the message space $\mathcal{M}$.

Honest Senders. On activation of the honest sender $\mathcal{P}_{S_i}$, where $i \in [n]$, simulate by executing the code of the honest sender on input the simulated message $\tilde{\mathtt{msg}}_j$ chosen uniformly at random, without re-introduction, from $\mathcal{M}_H$.

Extraction of the Inputs. Let L_{h^*-1} be the list produced by the malicious mixer $\mathcal{P}_{M_{h^*-1}}$. For any $j \in [n]$, decrypt $\mathtt{msg}_j \leftarrow\!\!\$\ \mathsf{Dec}(\mathsf{sk}, \mathtt{C}_{h^*-1,j})$ and if a decryption error occurs, or some of the mixer proofs π^j_{mx} is not valid, i.e. the event `Invalid` occurs, abort the simulation. If $\mathtt{msg}_j \notin \mathcal{M}_H$ then submit it as input to the ideal functionality $\mathcal{F}_{\text{Mix}}$.

First Honest Mixer. Simulate by computing L_{h^*} as a list of encryption of random (distinct) messages $\mathtt{H}_1, \ldots, \mathtt{H}_n$, simulating the proof of mixing $\pi^{h^*}_{\mathsf{mx}}$.

Verification Phase. Receive from the ideal mixer functionality $\mathcal{F}_{\text{Mix}}$ the sorted output $(\mathtt{msg}_i)_{i\in[n]}$. Sample a random permutation ζ_o and populate the list of outputs $M_o := (\mathtt{msg}_{o,i})_{i\in[n]}$ with $\mathtt{msg}_{o,\zeta_o(i)} \leftarrow \mathtt{msg}_i$.

We notice that there are some differences between $\mathbf{H}_{10}$ and the interaction of S with the ideal functionality $\mathcal{F}_{\text{Mix}}$. In particular, the hybrid defines the function ψ_{in} by setting a mapping between the inputs of the honest senders and the simulated ones, and, during the decryption phase, and uses ψ^{-1}_{in} to revert this change. S cannot explicitly set this mapping, because the inputs of the honest senders are sent directly to the functionality and are unknown to S. However, the simulator is implicitly defining the function ψ_{in} (and ψ^{-1}_{in}) since during the simulation chooses a simulated input $\tilde{\mathtt{msg}}_i$ for each honest sender and at decryption phase outputs the messages coming from the sorted list (given in output by the ideal functionality) which contains the inputs of the honest senders. □

6.6 A concrete Mix-Net protocol from RCCA-PKE

As already mentioned, to instantiate the blue-print protocol defined in Fig. 6.4 we need two main components: (1) a Rand lRCCA PKE scheme PKE and (2) a verify-then-decrypt protocol for such PKE.

6.6.1 Split PKE

We start by introducing the notion of Split Public-Key Encryption scheme. Informally, a Split PKE scheme is a special form of PKE scheme that extends and builds upon another PKE scheme. For example, CCA-secure PKE schemes à la Cramer-Shoup [CS98] can be seen as an extension of CPA-secure PKE schemes. We give the formal definition in the following.

Definition 6.6.1 (Split PKE)**.** *A split PKE scheme* PKE *is a tuple of seven randomized algorithms:*

$\mathsf{Setup}(1^\lambda)$: *upon input the security parameter* 1^λ *produces public parameters* pp*, which include the description of the message* ($\mathcal{M}$) *and two ciphertext spaces* ($\mathcal{C}_1, \mathcal{C}_2$).

$\mathsf{KGen}_A(\mathsf{pp})$: *upon input the parameters* pp*, outputs a key pair* $(\mathsf{pk}_A, \mathsf{sk}_A)$.

$\mathsf{KGen}_B(\mathsf{pp}, \mathsf{pk}_A)$: *upon inputs the parameters* pp *and a previously generated public key* pk_A*, outputs a key pair* $(\mathsf{pk}_B, \mathsf{sk}_B)$.

$\mathsf{Enc}_A(\mathsf{pk}_A, \mathtt{msg}; r)$: *upon inputs a public key* pk_A*, a message* $\mathtt{msg} \in \mathcal{M}$*, and randomness* r*, outputs a ciphertext* $\mathtt{C}_A \in \mathcal{C}_A$.

$\mathsf{Enc}_B(\mathsf{pk}_A, \mathsf{pk}_B, \mathtt{C}; r)$: *upon inputs a pair of public keys* $(\mathsf{pk}_A, \mathsf{pk}_B)$*, a ciphertext* $\mathtt{C} \in \mathcal{C}_A$*, and some randomness* r*, outputs a ciphertext* $\mathtt{C}_B \in \mathcal{C}_B$.

$\mathsf{Dec}_A(\mathsf{pk}_A, \mathsf{sk}_A, \mathtt{C})$: *upon inputs a secret key* sk_A *and a ciphertext* $\mathtt{C} \in \mathcal{C}_A$*, outputs a message* $\mathtt{msg} \in \mathcal{M}$ *or an error symbol* $\perp$.

$\mathsf{Dec}_B(\mathsf{pk}_A, \mathsf{pk}_B, \mathsf{sk}_A, \mathsf{sk}_B, \mathtt{C})$: *upon inputs secret keys* $\mathsf{sk}_A, \mathsf{sk}_B$ *and a ciphertext* $\mathtt{C} \in \mathcal{C}_B$*, outputs a message* $\mathtt{msg} \in \mathcal{M}$ *or an error symbol* $\perp$.

Moreover, we say that a split PKE scheme PKE splits on *a PKE scheme* $\mathsf{PKE}_A := (\mathsf{KGen}_A, \mathsf{Enc}_A, \mathsf{Dec}_A)$ *defined over message space* $\mathcal{M}$ *and ciphertext space* $\mathcal{C}_A$*, and we say that a split PKE scheme* PKE forms *a PKE* $\mathsf{PKE} := (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *defined over message space* $\mathcal{M}$ *and ciphertext space* $\mathcal{C}_B$ *where* $\mathsf{KGen}(\mathsf{pp})$ *is the algorithm that first runs* $\mathsf{pk}_A, \mathsf{sk}_A \leftarrow\!\$\ \mathsf{KGen}_A(\mathsf{pp})$*, then runs* $\mathsf{pk}_B, \mathsf{sk}_B \leftarrow\!\$\ \mathsf{KGen}_B(\mathsf{pp}, \mathsf{pk}_A)$ *and sets* $\mathsf{pk} := (\mathsf{pk}_A, \mathsf{pk}_B)$*,* $\mathsf{sk} := (\mathsf{sk}_A, \mathsf{sk}_B)$*, where* $\mathsf{Enc}(\mathsf{pk}, \mathtt{msg})$ *is the algorithm that outputs* $\mathsf{Enc}_B(\mathsf{pk}_A, \mathsf{pk}_B, \mathsf{Enc}_A(\mathsf{pk}_A, \mathtt{msg}; r); r)$ *and* $\mathsf{Dec} := \mathsf{Dec}_B$.

The correctness property is straightforward: a split PKE is correct if it forms a PKE that is correct in the standard sense. Our definition is general enough to capture a large class of schemes. We first note that any PKE scheme is trivially split: it suffices that Enc_B on input $\mathtt{C}$ outputs $\mathtt{C}$, and Dec_B runs Dec_A. A more natural (and less trivial) example is the above-cited Cramer-Shoup.

In this chapter, we will focus on PKE schemes that are Re-Randomizable and Verifiable. Since, as we noted above, any PKE can be parsed as a Split PKE, Re-Randomizability is captured by an additional algorithm $\mathsf{Rand}(\mathsf{pk}, \mathtt{C}; r)$ that takes as input a ciphertext $\mathtt{C}$ and outputs a new ciphertext $\hat{\mathtt{C}}$ (see Section 2.4.1).

As for the verifiability property, instead, there are three possible levels: (i) both the secret keys are required to verify a ciphertext, or (ii) only sk_A is needed, or (iii) no secret key is required at all. We refer to the third one as the *public* setting, while the other two are different flavors of a private/designated-verifier setting. We give the definition of (ii) in what follows.

Definition 6.6.2 (verifiable split PKE)**.** *A verifiable split PKE is a split PKE, as defined above, with an additional algorithm* $\mathsf{Verify}(\mathsf{pk}, \mathsf{sk}_B, \mathtt{C})$ *that takes as input the public key* pk*, the secret key* sk_B *and a ciphertext* $\mathtt{C} \in \mathcal{C}_B$ *and outputs* 1 *whenever* $\mathsf{Dec}_B(\mathsf{pk}, \mathsf{sk}, \mathtt{C}) \neq \perp$*, otherwise outputs* 0 *for invalid ciphertexts.*

The ideal functionality has as parameters a public-key encryption scheme $\mathsf{PKE} \coloneqq (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ and (implicit) public parameters pp. The functionality interacts with m parties $\mathcal{P}_i$ and with an adversary S.

Public Key. Upon activation on message $(\mathsf{KEY}, \mathsf{sid})$ from a party $\mathcal{P}_i$, $i \in [m]$, if $(\mathsf{sid}, \mathsf{pk}, \mathsf{sk})$ is not in the database sample $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{pp})$ and store the tuple $(\mathsf{sid}, \mathsf{pk}, \mathsf{sk})$ in the database and send $(\mathsf{KEY}, \mathsf{sid}, \mathsf{pk})$ to $\mathcal{P}_i$.

Decryption. Upon activation on $(\mathsf{DECRYPT}, \mathsf{sid}, C)$ from party $\mathcal{P}_i, i \in [m]$:

- If the tuple $(\mathsf{sid}, \mathsf{pk}, \mathsf{sk})$ does not exist in the database, ignore the message.
- Check that a tuple $(\mathsf{sid}, C, M_o, \mathcal{I})$, where $\mathcal{I} \subseteq [m]$, exists in the database; if so, update $\mathcal{I}$ including the index i. Else, parse C as $(\mathsf{C}_i)_i$ and compute the list $M_o \coloneqq (\mathsf{Dec}(\mathsf{sk}, \mathsf{C}_i))_{i \in [|C|]}$, and create the new entry $(\mathsf{sid}, C, M_o, \{i\})$ in the database.
- If $|\mathcal{I}|$ equals m, then send a public delayed output $(\mathsf{DECRYPT}, \mathsf{sid}, M_o)$ to the parties $\mathcal{P}_i$ for $i \in [m]$.

Figure 6.7: UC ideal functionality $\mathcal{F}_{\mathsf{Dec}}^{\mathsf{PKE}}$ for (n-out-n Threshold) Key-Generation and Decryption of PKE

6.6.2 A protocol for Verify-then-Decrypt for verifiable split PKE

We realize the Verify-then-Decrypt ideal functionality (see Section 6.4) needed to instantiate our Mix-Net protocol. Let PKE be a verifiable split PKE. We define in Fig. 6.9 the protocol Π_{VtDec} that realizes $\mathcal{F}_{\mathsf{VtDec}}$ in the $\mathcal{F}_{\mathsf{Com}}$-hybrid model. Before doing that, we need to assume an extra property for our verifiable split PKE, so we introduce the notion of linear key-homomorphism for a PKE.

Definition 6.6.3 (Linearly Key-Homomorphic PKE)**.** *A PKE* $\mathsf{PKE} \coloneqq (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *is linearly key-homomorphic if there exist PPT algorithms* $\mathsf{GenPK}, \mathsf{CheckPK}$ *and an integer* s *such that:*

- *The algorithm* $\mathsf{KGen}(\mathsf{pp})$*, where* pp *contains the description of a group of order* q*, first executes* $\mathsf{sk} \leftarrow\!\!\$\ \mathbb{Z}_q^s$*, and then produces the public key* $\mathsf{pk} \leftarrow\!\!\$\ \mathsf{GenPK}(\mathsf{sk})$*.*
- *The algorithm* GenPK *is linearly homomorphic in the sense that for any* $\mathsf{sk}_1, \mathsf{sk}_2 \in \mathbb{Z}_q^s$ *and* $\alpha \in \mathbb{Z}_q^s$ *we have* $\mathsf{GenPK}(\alpha \cdot \mathsf{sk}_1 + \mathsf{sk}_2) = \alpha \cdot \mathsf{GenPK}(\mathsf{sk}_1) + \mathsf{GenPK}(\mathsf{sk}_2)$*.*
- *The algorithm* $\mathsf{CheckPK}$ *on input the public key* pk *outputs a bit* b *to indicate if the public key belongs on the subgroup of* $\mathcal{PK}$ *spanned by* GenPK*. Namely, for any* pk *we have* $\mathsf{CheckPK}(\mathsf{pk}) = 1$ *if and only if* $\mathsf{pk} \in Im(\mathsf{GenPK}(\mathsf{pp}, \cdot))$*.*

Moreover, a split PKE PKE *is linearly key-homomorphic it forms a linearly key-homomorphic PKE, and it splits to a key-homomorphic PKE.*

It is not hard to verify that the key generation of a linearly key-homomorphic split PKE can be seen as sampling two secret vectors $\mathsf{sk}_A \in \mathbb{Z}_q^s$ and $\mathsf{sk}_B \in \mathbb{Z}_q^{s'}$ for $s, s' \in \mathbb{N}$ and then applying two distinct homomorphisms $\mathsf{GenPK}_A, \mathsf{GenPK}_B$ to derive the public key.

The functionality interacts with n parties $\mathcal{P}_i$ and an adversary S.

Commitment. Upon activation on message $(\mathsf{COMMIT}, \mathrm{sid}, \mathcal{P}_i, s)$ from a party $\mathcal{P}_i$, where $s \in \{0,1\}^*$, record the tuple $(\mathrm{sid}, \mathcal{P}_i, s)$ and send the public delayed output $(\mathsf{RECEIPT}, \mathrm{sid}, \mathcal{P}_i)$ to all the parties $\mathcal{P}_j$, $j \in [n], j \neq i$.

Opening. Upon activation on message $(\mathsf{OPEN}, \mathrm{sid}, \mathcal{P}_i)$ from a party $\mathcal{P}_i$, $i \in [n]$, proceed as follows: if the tuple $(\mathrm{sid}, \mathcal{P}_i, s)$ was previously recoded, then send the public delayed output $(\mathsf{OPEN}, \mathrm{sid}, \mathcal{P}_i, s)$ to all other parties $\mathcal{P}_j, j \in [n], j \neq i$. Otherwise halt.

Figure 6.8: UC ideal functionality $\mathcal{F}_{\mathsf{Com}}$ for (Single) Commitment.

Building Blocks. Let PKE be a split PKE that splits over PKE_A, consider the following building blocks:

1. An ideal functionality $\mathcal{F}_{\mathsf{Dec}}^{\mathsf{PKE}_A}$ for threshold decryption, as defined in Fig. 6.7, of PKE_A.
2. A single-sender multiple-receiver commitment ideal functionality $\mathcal{F}_{\mathsf{Com}}$ [CF01] for strings, as defined in Fig. 6.8.

We describe the protocol in Fig. 6.9. At a high level, the protocol works as follows. Each party $\mathcal{P}_i$ interacts with the ideal functionality $\mathcal{F}_{\mathsf{Dec}}$ to get the public key pk_A and, after that, samples the pair of keys $(\mathsf{pk}_B^i, \mathsf{sk}_B^i)$. The secret key is committed through the ideal functionality $\mathcal{F}_{\mathsf{Com}}$. After this step, the parties compute the final key pk_B as the sum of all their input public key shares. To verify the ciphertexts C_V, the parties reveal their secret key shares sk_B^i, verify that all the keys are consistent, and locally verify the ciphertexts. Finally, to decrypt the ciphertexts C_D, the parties invoke $\mathcal{F}_{\mathsf{Dec}}$ after checking that $C_D \subseteq C_V$.

Theorem 6.6.1. *Let* PKE *be a verifiable split PKE that is linearly key-homomorphic, let* f *be the leakage function that on input* $\mathsf{sk} := (\mathsf{sk}_A, \mathsf{sk}_B)$ *outputs* sk_B*. The protocol* $\Pi_{\mathsf{VtDec}}^{\mathsf{PKE}}$ *described in Fig. 6.9 UC-realizes the functionality* $\mathcal{F}_{\mathsf{VtDec}}^{\mathsf{PKE},f}$ *described in Fig. 6.3 with setup assumptions* $\mathcal{F}_{\mathsf{Dec}}^{\mathsf{PKE}_A}$ *and* $\mathcal{F}_{\mathsf{Com}}$*.*

Proof. We now prove that there exists a simulator S such that no PPT environment Z can distinguish an interaction with the real protocol from an interaction with S and the ideal functionality $\mathcal{F}_{\mathsf{VtDec}}$.

Simulator S.

Public Key. The simulator S receives in input from Z the set of corrupted parties, and receives from $\mathcal{F}_{\mathsf{VtDec}}$ the public key pk that is parsed as the tuple $(\mathsf{pk}_A, \mathsf{pk}_B)$. S gets to see the secret key shares of the corrupted parties when they send the message $(\mathsf{COMMIT}, \mathsf{sid}, \mathsf{sk}_B^i)$. Let h^* be the index of an honest party. S samples at random the secret keys sk_B^i for all honest parties $\mathcal{P}_i$, with $i \neq h^*$, from which can honestly compute the corresponding public keys through GenPK. As for the h^*-th party, S checks if $\forall j \neq h^* : \mathsf{CheckPK}(\mathsf{pk}_A, \mathsf{pk}_B^j) = 1$. If so it computes directly the public key $\mathsf{pk}_B^{h^*} := \mathsf{pk}_B - \sum_{i \neq h^*} \mathsf{pk}_B^i$, else it samples $\mathsf{sk}_B^{h^*}$ and computes the corresponding public key.

The party $\mathcal{P}_i$ executes the following commands:

Public Key. Upon activation on message:

leftmargin=0cm $(\mathsf{KEY}, \mathsf{sid})$ from the environment, forward the message to $\mathcal{F}_{\mathsf{Dec}}^{\mathsf{PKE}_A}$.

leftmargin=0cm $(\mathsf{KEY}, \mathsf{sid}, \mathsf{pk}_A)$ from $\mathcal{F}_{\mathsf{Dec}}^{\mathsf{PKE}_A}$ proceed as below:

1. Sample $\mathsf{sk}_B^i \leftarrow\!\!\$\ \mathbb{Z}_q^s$ compute $\mathsf{pk}_B^i \leftarrow \mathsf{GenPK}(\mathsf{sk}_B^i)$.
2. Commit the secret key sk_B^i through the ideal functionality $\mathcal{F}_{\mathsf{Com}}$, i.e. send the message $(\mathsf{COMMIT}, \mathsf{sid}, \mathsf{sk}_B^i)$ to the functionality $\mathcal{F}_{\mathsf{Com}}$.

leftmargin=0cm $(\mathsf{RECEIPT}, \mathsf{sid}, \mathcal{P}_j)$ from all $j \in [m]$ broadcast $(\mathsf{KEY}, \mathsf{sid}, i, \mathsf{pk}_B^i)$.

When the parties have sent their public key shares, compute $\mathsf{pk}_B := \sum_i \mathsf{pk}_B^i$ and abort if $\exists i : \mathsf{CheckPK}(\mathsf{pk}_A, \mathsf{pk}_B^i) = 0$ else output $(\mathsf{KEY}, \mathsf{sid}, \mathsf{pk})$.

Verify then Decrypt. Upon activation on message:

- $(\mathsf{VTDEC}, \mathsf{sid}, C_V, C_D)$ send $(\mathsf{OPEN}, \mathsf{sid}, \mathcal{P}_i)$ to $\mathcal{F}_{\mathsf{Com}}$ and broadcast $(\mathsf{VTDEC}, \mathsf{sid}, C_V, C_D)$ to the other parties.
- $(\mathsf{OPEN}, \mathsf{sid}, \mathcal{P}_j, \mathsf{sk}_B^j)$ for all $i \in [m]$ compute $\mathsf{sk}_B := \sum_i \mathsf{sk}_B^j$ and assert that $\mathsf{GenPK}_B(\mathsf{sk}_B) \stackrel{?}{=} \mathsf{pk}_B$ and that all parties broadcast the same lists C_V and C_D. Parse C_V as $(\mathsf{c}_V^i)_{i \in |\mathcal{C}_V|}$, compute $\forall j : b_j \leftarrow \mathsf{Verify}(\mathsf{pk}, \mathsf{sk}_B, \mathsf{c}_V^j)$.
  If $C_D \not\subseteq C_V$ or $\exists i : b_i = 0$ return $(\mathsf{DECRYPT}, \mathsf{sid}, \vec{b}, ())$ else send $(\mathsf{DECRYPT}, \mathsf{sid}, C_D)$ to $\mathcal{F}_{\mathsf{Dec}}^{\mathsf{PKE}_A}$ and upon receipt of $(\mathsf{DECRYPT}, \mathsf{sid}, M_o)$, output $(\mathsf{DECRYPT}, \mathsf{sid}, \vec{b}, M_o)$

Figure 6.9: Our protocol $\Pi_{\mathsf{VtDec}}^{\mathsf{PKE}}$.

Verification. When all the parties have sent the message $(\mathsf{OPEN}, \mathsf{sid}, \mathcal{P}_i)$ to the commitment functionality $\mathcal{F}_{\mathsf{Com}}$, the simulator receives the leakage $(\mathsf{sid}, \mathsf{sk}_B)$ from $\mathcal{F}_{\mathsf{VtDec}}^{\mathsf{PKE},f}$, it computes the secret key for party $\mathcal{P}_{h^*}$, i.e. it computes $\mathsf{sk}_B^{h^*} := \mathsf{sk}_B - \sum_{i \neq h^*} \mathsf{sk}_B^i$. From this point on, the simulation becomes trivial since the simulator follows the protocol, and can easily verify and decrypt all the ciphertexts by interacting with the ideal functionality $\mathcal{F}_{\mathsf{VtDec}}$.

We observe that the inputs simulated for the honest parties $\mathcal{P}_i$, for $i \neq h^*$, are perfectly simulated since S chooses uniformly at random the matrices and the vectors for the secret keys sk_B^i. The public key for the h^*-th party is chosen dependently of the message of the corrupted parties. In particular, if one of the corrupted parties sends an invalid public key the h^*-th mixer follows the specification of the protocol, thus the simulation is perfect; if all the public keys are valid, the public key of h^*-th party is chosen as a function of the previously chosen keys and the public key given in input to the simulator. This is distributed identically to a real execution of the protocol: the only difference is that S computes the random public key, while in the real execution the party $\mathcal{P}_{h^*}$ would choose at random their secret key and then project it to compute the corresponding public key, but this difference is only syntactical. In the next steps, the simulation is perfect since it proceeds exactly as in the real protocol. □

6.6.3 Our concrete verifiable split PKE

In this section, we show that the Rand-PKE in [FFHR19] has all the properties needed to instantiate our protocol Π_{Mix}. In particular, in Fig. 6.10 we parse their PKE as a split PKE, and we prove that the scheme is lRCCA w.r.t. the leakage function f such that $f(\mathsf{sk}) := \mathsf{sk}_B$, and that the scheme is linearly key-homomorphic. Moreover, we show a checksum-admissible relation $\mathcal{R}_{\mathsf{mx}}$ w.r.t. the PKE, we show how to instantiate the ABO perfect sound tag-based NIZK $\mathsf{NIZK}_{\mathsf{mx}}$ using the Kiltz-Wee quasi-adaptive (QA) NIZK [KW15], and we instantiate the NIZK $\mathsf{NIZK}_{\mathsf{sd}}$ using GS-Proofs.

The schemes in [FFHR19] are proven secure under a decisional assumption that we briefly introduce here. Let ℓ, k be two positive integers. We call $\mathcal{D}_{\ell,k}$ a matrix distribution if it outputs (in probabilistic polynomial time, with overwhelming probability) matrices in $\mathbb{Z}_q^{\ell \times k}$.

Definition 6.6.4 (Matrix Decisional Diffie-Hellman Assumption in $\mathbb{G}_\gamma$, [EHK+13])**.** *The $\mathcal{D}_{\ell,k}$-MDDH assumption holds if for all non-uniform PPT adversaries $\mathcal{A}$,*

$$\left|\Pr\left[\mathcal{A}(\mathcal{G}, [\vec{A}]_\gamma, [\vec{A}\vec{w}]_\gamma) = 1\right] - \Pr\left[\mathcal{A}(\mathcal{G}, [\vec{A}]_\gamma, [\vec{z}]_\gamma) = 1\right]\right| \in \mathsf{negl}(\lambda),$$

where the probability is taken over $\mathcal{G} := (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathcal{P}_1, \mathcal{P}_2) \leftarrow \mathsf{GroupGen}(1^\lambda)$, $\vec{A} \leftarrow\!\!\$\, \mathcal{D}_{\ell,k}$, $\vec{w} \leftarrow\!\!\$\, \mathbb{Z}_q^k$, $[\vec{z}]_\gamma \leftarrow\!\!\$\, \mathbb{G}_\gamma^\ell$ and the coin tosses of adversary $\mathcal{A}$.

Theorem 6.6.2. *The split PKE PKE described in Fig. 6.10 is linearly key-homomorphic and lRCCA-secure w.r.t. f such that $f(\mathsf{sk}) := \mathsf{sk}_B$ under the $\mathcal{D}_{k+1,k}$-MDDH assumption.*

Proof. To show that the scheme PKE of [FFHR19] is linear key-homomorphic, we briefly sketch that there exist the algorithms GenPK and $\mathsf{CheckPK}$ satisfying the property. The KGen algorithm samples the secret material $\mathsf{sk} := (\vec{a}, \vec{f}, \vec{F}, \vec{g}, \vec{G})$, consisting of vectors and matrices of elements in $\mathbb{Z}_q$. Then, in order to compute the public key pk, the secret key is projected: GenPK produces some matrices $([\vec{f}_j]_T, [\vec{F}_j]_1, [\vec{g}_j]_T, [\vec{G}_j]_2, [\vec{H}_j]_1, [\vec{I}_j]_2)$. It is also immediate to check that,

$\underline{\mathsf{KGen}_A(\mathsf{pp})}$

$\vec{D} \leftarrow\!\!\$\ \mathcal{D}_k;\quad \vec{a} \leftarrow\!\!\$\ \mathbb{Z}_q^{k+1}$

$\vec{D}^* \leftarrow (\vec{D}^\top, (\vec{a}^\top \vec{D})^\top)^\top$

$\mathsf{sk}_A \leftarrow \vec{a};\quad \mathsf{pk}_A \leftarrow ([\vec{D}]_1, [\vec{a}^\top \vec{D}]_1)$

return $(\mathsf{pk}_A, \mathsf{sk}_A)$

$\underline{\mathsf{KGen}_B(\mathsf{pp}, \mathsf{pk}_A)}$

$\vec{E} \leftarrow\!\!\$\ \mathcal{D}_k;\quad \vec{f}, \vec{g} \leftarrow\!\!\$\ \mathbb{Z}_q^{k+1}$

$\vec{F} \leftarrow\!\!\$\ \mathbb{Z}_q^{k+1\times k+1}, \vec{G} \leftarrow\!\!\$\ \mathbb{Z}_q^{k+1\times k+2}$

$\mathsf{sk}_B \leftarrow (\vec{f}, \vec{g}, \vec{F}, \vec{G})$

$\mathsf{pk}_B \leftarrow ([\vec{E}]_2, [\vec{f}^\top \vec{D}]_T, [\vec{F}^\top \vec{D}]_1,$
$[\vec{g}^\top \vec{E}]_T, [\vec{G}^\top \vec{E}]_2, [\vec{G}\vec{D}^*]_1, [\vec{F}\vec{E}]_2)$

return $(\mathsf{pk}_B, \mathsf{sk}_B)$

$\underline{\mathsf{Dec}_A(\mathsf{pk}_A, \mathsf{sk}_A, \mathsf{C} = [\vec{x}]_1)}$

return $[p]_1 - [\vec{a}^\top \vec{u}]_1$

$\underline{\mathsf{Enc}_A(\mathsf{pk}_A, [\mathtt{msg}]_1; \vec{r})}$

$[\vec{u}]_1 \leftarrow [\vec{D}]_1 \cdot \vec{r};\quad [p]_1 \leftarrow [\vec{a}^\top \vec{D}]_1 \cdot \vec{r} + [\mathtt{msg}]_1$

return $([\vec{u}^\top]_1, [p]_1)^\top$

$\underline{\mathsf{Enc}_B(\mathsf{pk}, \mathsf{C} = [\vec{x}]_1; (\vec{r}, \vec{s}))}$

$[\vec{v}]_2 \leftarrow [\vec{E}]_2 \cdot \vec{s}$

$[\pi_1]_T \leftarrow [\vec{f}^\top \vec{D}]_T \cdot \vec{r} + e([\vec{F}^\top \vec{D}]_1 \cdot \vec{r}, [\vec{v}]_2)$

$[\pi_2]_T \leftarrow [\vec{g}^\top \vec{E}]_T \cdot \vec{s} + e([\vec{x}]_1, [\vec{G}^\top \vec{E}]_2 \cdot \vec{s})$

$[\pi]_T \leftarrow [\pi_1]_T + [\pi_2]_T$

return $([\vec{x}]_1, [\vec{v}]_2, [\pi]_T)$

$\underline{\mathsf{Dec}_B(\mathsf{pk}, \mathsf{sk}, \mathsf{C} = ([\vec{x}]_1, [\vec{v}]_2, [\pi]_T))}$

$[\pi_1]_T \leftarrow [(\vec{f} + \vec{F}\vec{v})^\top \vec{u}]_T$

$[\pi_2]_T \leftarrow [(\vec{g} + \vec{G}\vec{x})^\top \vec{v}]_T$

if $[\pi]_T \neq [\pi_1]_T + [\pi_2]_T$ **return** $\perp$

else return $\mathsf{Dec}_A(\mathsf{sk}_A, [\vec{x}]_1)$

$\underline{\mathsf{Rand}(\mathsf{pk}, \mathsf{C} = ([\vec{x}]_1, [\vec{v}]_2, [\pi]_T))}$

parse $[\vec{x}]_1$ as $([\vec{u}^\top]_1, [p]_1)^\top,\quad \hat{\vec{r}}, \hat{\vec{s}} \leftarrow\!\!\$\ \mathbb{Z}_q^k$

$[\hat{\vec{x}}]_1 \leftarrow [\vec{x}]_1 + [\vec{D}^*]_1 \cdot \hat{\vec{r}},\quad [\hat{\vec{v}}]_2 \leftarrow [\vec{v}]_2 + [\vec{E}]_2 \cdot \hat{\vec{s}}$

$[\hat{\pi}_1]_T \leftarrow [\vec{f}^\top \vec{D}]_T \cdot \hat{\vec{r}} + e([\vec{F}^\top \vec{D}]_1 \cdot \hat{\vec{r}}, [\hat{\vec{v}}]_2) + e([\vec{u}]_1, [\vec{F}\vec{E}]_2 \cdot \hat{\vec{s}})$

$[\hat{\pi}_2]_T \leftarrow [\vec{g}^\top \vec{E}]_T \cdot \hat{\vec{s}} + e([\hat{\vec{x}}]_1, [\vec{G}^\top \vec{E}]_2 \cdot \hat{\vec{s}}) + e([\vec{G}\vec{D}^*]_1 \cdot \hat{\vec{r}}, [\vec{v}]_2)$

$[\hat{\pi}] \leftarrow [\pi]_T + [\hat{\pi}_1]_T + [\hat{\pi}_2]_T$

return $([\hat{\vec{x}}]_1, [\hat{\vec{v}}]_2, [\hat{\pi}]_T)$

Figure 6.10: The Split RCCA-secure Scheme. pp include the description of a bilinear group.

for any $\alpha \in \mathbb{Z}_q$ and any two secret keys $\mathsf{sk}_1, \mathsf{sk}_2$, the linear homomorphic property holds, i.e. $\mathsf{GenPK}(\alpha \cdot \mathsf{sk}_1 + \mathsf{sk}_2) = \alpha \cdot \mathsf{GenPK}(\mathsf{sk}_1) + \mathsf{GenPK}(\mathsf{sk}_2)$. To verify that $\mathsf{CheckPK}(\mathsf{pk})$, instead, it is sufficient to check the following pairing equations:

$$e([\vec{\tilde{H}}]_1, [\vec{E}]_2) = e([\vec{D}^*]_1, [\vec{\tilde{G}}_j]_2)$$
$$e([\vec{D}]_1, [\vec{\tilde{I}}]_2) = e([\vec{\tilde{F}}]_1, [\vec{E}]_2)$$

We now prove that the scheme is lRCCA-secure w.r.t. the function f, where $f(\mathsf{sk}) \coloneqq \mathsf{sk}_B$. We briefly recall the proof strategy from [FFHR19], and we only highlight the main differences.

- The hybrid $\mathbf{H}_1$ computes the challenge ciphertext using the secret key. Namely, $p^* \leftarrow \vec{a}^\top \vec{u}^* + \mathtt{msg}_b$ and similarly the proofs are $\pi_1^* \leftarrow \vec{f}^\top \vec{D}\vec{r} + \vec{v}^\top \vec{F}^\top \vec{D}\vec{r}\vec{F}$ and $\pi_2^* \leftarrow \vec{g}^\top \vec{E}\vec{s} + \vec{x}^\top \vec{G}^\top \vec{E}\vec{s}$. This hybrid is equivalent to the lRCCA experiment.

- The hybrid $\mathbf{H}_2$ samples for the challenge ciphertext $\vec{u}^*$ and $\vec{v}^*$ uniformly at random from $\mathbb{Z}_q^{k+1}$. The hybrids $\mathbf{H}_1$ and $\mathbf{H}_2$ are computationally indistinguishable. This follows by applying the $\mathcal{D}_k$-MDDH Assumption on $[\vec{D}, \vec{u}^*]_1$ in $\mathbb{G}_1$ and $[\vec{E}, \vec{v}^*]_2$ in $\mathbb{G}_2$. Notice that the reduction can sample sk_2 itself and easily reveal sk_2 to the adversary.

From now on, we prove that each pair of consecutive hybrids is statistically close. In particular, this means that the hybrids (and, in principle, also the adversary) are allowed to run in unbounded time.

- The hybrid $\mathbf{H}_3$ adds the two decryption rules, if $\vec{u} = \vec{D}\vec{r}$ uses $\vec{r}$ to compute π_1 and $\mathtt{msg}$ at decryption time, similarly, if $\vec{v} = \vec{E}\vec{s}$ then uses $\vec{r}$ to compute π_2. It is easy to see that this hybrid is equivalent to the previous by the linearity of the decryption procedure.

- The hybrid $\mathbf{H}_4$ adds another decryption rule, if $\vec{u} \notin \mathit{span}(\vec{D})$ and $\vec{v} - \vec{v}^* \notin \mathit{span}(\vec{E})$ then the decryption oracle returns $\perp$ to the adversary. This is the main core of the technique of [FFHR19]. In particular, they show that the probability that the proof π of the queried ciphertext is valid when the condition holds is $O(1/q)$. By inspection of their reduction, we notice that the reduction stops the simulation of the hybrid $\mathbf{H}_4$ as soon as a decryption query triggers the event in the newly added decryption rule. In particular, it means that the reduction does not have to simulate for the adversary the leakage value $f(\mathsf{sk}) = \mathsf{sk}_2$. In other words, the reduction [FFHR19] works exactly the same also in our leakage resilient experiment because it stops the adversary before the latter may receive the leaked value.

- The hybrid $\mathbf{H}_5$, similarly to the previous hybrid, adds a new the decryption rule that if $\vec{v} \notin \mathit{span}(\vec{E})$ and $\vec{x} - \vec{x}^* \notin \mathit{span}(\vec{D}^*)$ then the decryption oracle returns $\perp$ to the adversary. The proof for this hybrid is almost identical to the proof for the previous hybrid.

- The hybrid $\mathbf{H}_6$ adds a new decryption rule that if $\vec{x} - \vec{x}^* \in \mathit{span}(\vec{D}^*)$ and $\vec{v} - \vec{v}^* \in \mathit{span}(\vec{E})$ then the proof is verified in an alternative way. In particular, let $\vec{\tilde{r}}, \vec{\tilde{s}}$ be such that $\vec{x} - \vec{x}^* = \vec{\tilde{x}} = \vec{D}\vec{\tilde{r}}$ it computes π' as a re-randomization of the proof of the challenge ciphertext π^* using randomness $\vec{\tilde{r}}$ and $\vec{\tilde{s}}$ and if $\pi' \neq \pi$ the decryption oracle returns $\perp$ to the adversary. The proof of this hybrid follows the correctness of the re-randomization algorithm for the PKE.

- The hybrid $\mathbf{H}_7$ is the same as the previous, but it never uses the secret key material sk to decrypt. This hybrid is only syntactically different from the previous; in fact, by the decryption rules added in the previous hybrids also the $\mathbf{H}_6$ would never use the secret key material for the decryption query.

At this point, we can show that $\mathbf{H}_6$ is independent of the challenge bit even if the adversary additionally gets to see the leakage sk_2. Notice that only dependence on b is given by $p^* = \vec{a}^\top \vec{u}^* + \mathtt{msg}_b$. Moreover, b is independent of sk_2 thus, even leaking this value the view of the adversary is independent of the challenge bit. □

6.6.4 Putting all together

We can instantiate the ABO Perfect Hiding NIZK proof of membership $\mathsf{NIZK}_{\mathsf{mx}}$ using Groth-Sahai proofs [EG14]. In particular, notice that the necessary tag-space for $\mathsf{NIZK}_{\mathsf{mx}}$ is the set $[m]$ which in typical scenarios is a constant small number (for example 3 mixers). Thus, we can instantiate the tag-based ABO Perfect Hiding $\mathsf{NIZK}_{\mathsf{mx}}$ by considering an Init algorithm that samples m different common reference strings $(\mathsf{crs}_i)_{i\in[m]}$, the prover algorithm (resp. the verify algorithm) on tag j invokes the GS prover algorithm (resp. verifier algorithm) with input the common reference string crs_j. We can instantiate the tag-based ABO Perfect Sound NIZK $\mathsf{NIZK}_{\mathsf{sd}}$ using the technique presented in the full version of [FFHR19] (see Section 6.2 for more details). By the universal composability theorem, once we compose the protocol Π_{Mix} from Fig. 6.6 and Π_{VtDec} from Fig. 6.9 we obtain a protocol with setup assumption $\mathcal{F}_{\mathsf{Dec}}$, $\mathcal{F}_{\mathsf{Com}}$ and $\mathcal{F}_{\mathrm{CRS}}$. The first ideal functionality can be implemented using classical approaches (for example, see Benaloh [Ben06]). Briefly, the mixers can compute the shares of the public key $[\vec{a}^\top \vec{D}]_1$ for KGen_A as in Fig. 6.10 and prove the knowledge of the secret key share $\vec{a}^{(i)}$ where $\vec{a} = \sum_i \vec{a}^{(i)}$, to obtain UC security in the malicious setting against static corruptions we can use an ABO Perfect Hiding NIZK proof system for this step. At decryption time, the mixers can compute a batched zero-knowledge proof of knowledge for "encryption of zero", they can use a NIZK proof of membership and, for UC security, it is sufficient for such proofs to be adaptive perfect sound.

Auditability. Here we sketch the auditability of our protocol. Roughly speaking, a protocol Π is *auditable* if there exists a PT algorithm Audit that on input a transcript τ and an output y output 1 if and only if the execution of the protocol that produces the transcript τ ends up with the parties outputting y. We focus on the auditability of the protocol obtained composing Π_{Mix} from Fig. 6.6 and Π_{VtDec} from Fig. 6.9. The auditing algorithm, given a transcript of Π_{VtDec} can reconstruct the secret key sk_2 and can check that $\mathsf{Verify}(\mathsf{sk}_2, \mathsf{C}_{i,j}) = 1$ for all $i \in [m]$ and $j \in [n]$ moreover it checks that all the NIZK proofs verify. The checks performed guarantee that the protocol execution resulting to the transcript did not abort, moreover, the auditability is guaranteed by the correctness of the protocol. Finally, we notice that the protocol for $\mathcal{F}_{\mathsf{Dec}}$ sketched in the previous section is auditable (see [Ben06]).

Efficiency. We analyze the efficiency of the protocol obtained composing Π_{Mix} and Π_{VtDec}, and we consider the most efficient instantiation of the scheme in [FFHR19] based on SXDH assumption, i.e. for $k = 1$. Let E_1, E_2 (resp. E_T) be the cost of a multiplication in groups $\mathbb{G}_1$ and $\mathbb{G}_2$ (resp. exponentiation in $\mathbb{G}_T$), and let P be the cost of computing a bilinear pairing. We give an intuition on how much the protocol scales when a mixer is given N processors and may make use of parallelism. We compare our results with the Mix-Net protocol of [FFHR19].

In our protocol Π_{Mix}, each mixer re-randomizes a list of n ciphertexts which requires $n(7E_1 + 7E_2 + 2E_T + 9P)$, and additionally computes a proof π_{mx} for the sumcheck relation $\mathcal{R}_{\mathsf{mx}}$ which requires n additions in $\mathbb{Z}_q$ and $6E_1 + 8E_2$. Re-randomization of a ciphertext in the list does not depend on other ciphertexts in the list, so the parallel cost is $\frac{n}{N}(7E_1 + 7E_2 + 2E_T + 9P)$. Additionally, the mixers verify all the sumcheck NIZK proofs, which requires $3nm$ additions in $\mathbb{G}_1$ and around 8 pairings. The parallel cost is $\frac{8m}{N}$ pairings plus $\log_N(3n)\frac{m}{N}$ additions.

In the protocol Π_{VtDec}, each mixer sends a commitment of their secret key share, which requires a UC-commitment for the elements of the secret key sk, and receives commitments of secret key shares of the other $m - 1$ mixers. Additionally, the mixers derive the public key shares, using GenPK, this corresponds to the cost of generating m times a key pk_B^i and requires $m(4E_T + 6E_1 + 6E_2)$. Finally, each mixer needs to verify the $n \cdot m$ ciphertexts produced in the protocol execution of the last list which requires $n(m - 1)(6E_1 + 4E_2 + 4P)$.

In the protocol of [FFHR19] the public key shares pk_B^i (and not the secret ones) are committed using an equivocable commitment and an ABO NIZK proof (which can be seen as a UC-secure commitment against static corruption). The parallel cost of re-randomize their ciphertexts is $\frac{n}{N}(36E_1 + 45E_2 + 6E_T + 5P)$, while the cost of verifying the ciphertexts and decrypting the last list is equal to $\frac{nm}{N}36P + \frac{m}{N}(2E_1 + 50P)$. In comparison, our approach allows saving at least $\frac{n}{N}(30E_1 + 39E_2 + 36P)$ cryptographic operations, where we recall that n is the number of shuffled ciphertexts.

Chapter 7

Re-Randomizable PKE meets Tight security

This chapter is extracted from "Almost Tightly-Secure Re-Randomizable and Replayable CCA-secure Public Key Encryption", published in PKC 2023.

7.1 Introduction

Security against chosen-ciphertext attacks (CCA) is considered to be the standard notion of security for PKE schemes. This security definition, formulated by Rackoff and Simon [RS92], is elegant and easy to understand, and it has shown, by any means, to withstand the test of time.

Replayable and Re-Randomizable CCA security. Canetti, Krawczyk and Nielsen [CKN03] pointed out that CCA security is not necessary for implementing secure channels. They showed that "replayable chosen-ciphertext" (RCCA) security suffices for secure channels, and might in fact allow for more efficient instantiations. Subsequently, Groth [Gro04] showed that RCCA PKE schemes (called Rand-RCCA secure) can have re-randomizable ciphertexts. Specifically, Groth constructed a scheme with a ciphertext re-randomization procedure that, given a ciphertext as input, produces a fresh and unlinkable ciphertext which decrypts to the same message. Such a re-randomization procedure opens the door for applications that require secure communication *and* anonymity.

For instance, PKE schemes that are re-randomizable and RCCA-secure enable anonymous and secure message transmissions (see Prabhakaran and Rosulek [PR07]), Mix-Nets (see Faonio *et al.* [FFHR19] and Pereira and Rivest [PR17]), Controlled Functional Encryption (see Naveed *et al.* [NAP+14]), and one-round message-transmission protocols with reverse firewalls (see Dodis, Mironov, and Stephens-Davidowitz [DMS16]).

Tight Security. Yet another criticism to the original definition of CCA security is that while the definition postulates that the message underlying *one single* ciphertext remains protected even under CCA attacks, in the real world, a PKE scheme is used to protect a large amount of ciphertexts from possibly many users.

Now, it is well-known that security for one single ciphertext implies, through a hybrid argument, security for many ciphertexts and many users. However, it is unclear how much *concrete* security a PKE scheme really offers when it is used in the wild. This question,

initially posed by Bellare, Boldyreva and Micali [BBM00] created a fruitful area of research which investigates how tight the security of an encryption scheme translates to the trust that we have with respect to the cryptographic assumption that it relies on.

In more detail, a tight security reduction ensures that for any attack on the PKE scheme, there exists an attack on the assumption that is similar both in terms of complexity (i.e. the running time, the space required, etc.) and success probability. Thus, in the setting of tight security reductions, the number of ciphertexts considered by the security definition matters.

By now, many CCA-secure PKE schemes have been proved to have tight security in the multi-ciphertext and multi-user setting: some notable examples are the works of [GHKW16, GHK17, HLLG19, Hof17, LJYP14, LPJY15]. However, tight security in the context of Rand-RCCA security has not been studied, although in particular the above Rand-RCCA use cases feature many ciphertexts or users.

7.1.1 Our Contributions

We initiate the study of tight security for Rand-RCCA secure PKE schemes in the multi-ciphertext and multi-user setting. Our main contributions are a new security definition for RCCA security in multi-ciphertext and multi-user setting (hereafter, mRCCA security), and a Rand-mRCCA PKE scheme whose mRCCA security (almost[1]) tightly reduces to the $\mathcal{D}_d$-MDDH assumption in symmetric (a.k.a. type-1) pairing groups. Moreover, as an application, we revise the protocol for universally composable MixNet based on Rand-RCCA PKE from [FFHR19]. In the following paragraphs, we elaborate more about each of the contributions.

Multi-user Multi-ciphertext RCCA security. In the security experiment of the (single-ciphertext) RCCA security notion, the decryption oracle, called "guarded decryption oracle", can be queried on any ciphertext, including the challenge ciphertext. However, when decryption leads to one of the challenge messages $(\mathtt{msg}_0, \mathtt{msg}_1)$, the oracle answers with a special symbol $\diamond$ (meaning "same"). As a warm-up, consider a trivial extension to the case of (single-user) multi-ciphertext RCCA security where the attacker is given:

- an encryption oracle that, on input a pair of messages $\mathtt{msg}_0, \mathtt{msg}_1$, returns some valid encryption of $\mathtt{msg}_b$ where b is the challenge bit,
- and a guarded decryption oracle that, on input a ciphertext $\mathtt{C}$, returns a message $\mathtt{msg}$, or the special indexed symbol $\diamond_j$ if $\mathtt{C}$ corresponds to an encryption of a message that was given as input to the encryption oracle as j-th query.

We notice that this trivial extension of RCCA security to multiple ciphertexts is impossible to achieve. Namely, consider the following generic attacker $\mathcal{A}$ that makes three queries to the encryption oracle:

1. $\mathcal{A}$ sends $(\mathtt{msg}_1, \mathtt{msg}_2)$, and receives back $\mathtt{C}_A$;
2. sends $(\mathtt{msg}_2, \mathtt{msg}_3)$, and receives back $\mathtt{C}_B$;
3. sends $(\mathtt{msg}_3, \mathtt{msg}_1)$, and receives back $\mathtt{C}_C$.

[1] As most of the tightly-secure schemes, the security reduction suffers from a small multiplicative loss that is however independent of the number of uses of the scheme.

$\mathcal{A}$ now queries the decryption oracle with $\mathtt{C}_C$. If the bit b is 0, the decryption oracle returns $\diamond_2$; if b is 1, the decryption oracle returns $\diamond_1$.

Yet another natural extension of the single-ciphertext RCCA security notion to the multi-ciphertext setting is to consider a guarded decryption oracle that upon input a ciphertext $\mathtt{C}$ either returns a message or the special symbol $\diamond$, but without notifying the adversary of which index j triggered the special symbol. Even if this definition avoids the attack described above, it is not as convenient as we would like it to be. Roughly speaking, the guarded decryption oracle reveals to the adversary that the queried ciphertext is a replay attack, but it doesn't tell which ciphertext was replayed; therefore, the larger the number of challenge ciphertexts, the less informative the output of the guarded decryption oracle will be. In particular, this definition is not sufficient for our MixNet application.

"In medio stat virtus", as the saying goes: the definition we propose is weaker than the first attempted (yet impossible to achieve) definition, but stronger than the above-mentioned definition. To build some intuition, in an equivalent version of the single-ciphertext RCCA security definition, the guarded decryption oracle would output the minimal set of messages that the queried ciphertext could decrypt to and such that such set does not trivially break the RCCA security definition: namely, if the ciphertext is a replay attack then the oracle replies with the set of challenge messages $\{\mathtt{msg}_0, \mathtt{msg}_1\}$, otherwise with a message $\mathtt{msg}' \notin \{\mathtt{msg}_0, \mathtt{msg}_1\}$. We take a similar approach in our (multi-user) multi-ciphertext RCCA definition. The guarded decryption oracle outputs the minimal set of messages that the ciphertext could decrypt to without trivially breaking security. This set of messages includes all the pairs of challenge messages for which at least one of them is equal to the decryption of the queried ciphertext.

To support the claim that our definition is indeed the most natural extension of RCCA to the multi-ciphertext setting, we prove that the simulation-based notion for RCCA security from [CKN03] is tightly implied by our mRCCA security notion. In a nutshell, we show that the (computational variational) distance between the view in the ideal world and in the real world is bounded by the advantage of an adversary with the same computational resources as the environment in the multi-ciphertext RCCA security game. We elaborate on the details in Section 7.5.

A Tightly-Secure Rand-mRCCA PKE scheme. Our starting points are the recent work of Faonio *et al.* [FFHR19] which is the state-of-the-art for Rand-RCCA PKE scheme, and the tightly-secure CCA PKE schemes based on the adaptive partitioning techniques of Hofheinz [Hof17] and Gay *et al.* [GHKP18]. Very briefly, the main idea of our construction is to encrypt the message similarly to [FFHR19], and additionally append a non-interactive proof of consistency for (part of) the ciphertext; the latter proof needs to have a (weak) form of simulation soundness property that can be obtained information-theoretically. Namely, using the notation of [Hof17], we append to the ciphertext a *benign proof* for the consistency of part of the ciphertext (which lies in a linear language) of a proof system that is statistically sound even when the adversary has oracle access to simulated proofs for a larger language that includes the disjunction of two linear spaces.

Extensions and applications. Following the strategy of [FFHR19] we show that our Rand-mRCCA PKE can be used to instantiate a PKE with the nice property of public-verifiable ciphertexts (pv-Rand-mRCCA PKE). We propose two pv-Rand-mRCCA PKE schemes: one based on the Matrix Diffie-Hellman Assumption (MDDH), and a second more efficient scheme based on a new MDDH assumption which we prove secure in the generic group model. As

an application of our framework, we show that we can plug a pv-Rand-mRCCA scheme into the MixNet protocol of [FFHR19]. Instantiating such protocol with our schemes, we obtain an (almost) *"tightly-secure"* MixNet protocol: namely a protocol, the first of its kind, whose security guarantees depend linearly on the number of mixer parties but only logarithmically on the number of mixed messages. To compare with the state of the art for MixNet protocols, we notice that the Bayer and Groth [BG12] proof of shuffle is based on the Fiat-Shamir transform applied to a multi-round Sigma protocol, thus the security reduction degrades with the number of rounds of the underlying Sigma-Protocol, while the proof of shuffle in the bilinear-paring setting of Fauzi *et al.* [FLSZ17] relies on new kinds of $\mathcal{D}_n$-KerMDH assumptions (proved to hold generically in the same paper) where n is the number of shuffled ciphertexts.

7.2 A Technical Overview of Our Results

To go from the rough idea described above to the actual scheme, we need to overcome two technical problems. The first problem is that our benign proof system needs to be re-randomizable (or, to better say, "malleable" as it needs to be able to re-randomize proofs of re-randomized statements), as we are aiming to construct a Rand-PKE scheme. We notice that none of the benign proof systems or affine notions we are aware of (such as [AJOR18, GHK17, GHKP18, Hof17]) are re-randomizable. To solve this problem, we introduce a new malleable proof system based on the work of Abdalla, Benhamouda and Pointcheval [ABP15], with the necessary security guarantees.

The second (and more challenging) technical problem is that we need to reconcile the adaptive partitioning technique with the Rand-RCCA technique of [FFHR19]. In particular, at the core of the adaptive partitioning technique there is a complex argument that shows that the decryption oracle can safely reject *ill-formed* ciphertexts even when the adversary can observe (many) ill-formed challenge ciphertexts. In some sense, these challenge ciphertexts are the only ill-formed ciphertexts that correctly decrypt, while all other ill-formed ciphertexts produced by the adversary do not. However, in our security proof the adversary can easily produce ill-formed ciphertexts that correctly decrypt, simply by re-randomizing challenge ciphertexts.

In more detail, the adaptive partitioning technique moves the challenge ciphertexts back and forth between two different linear spaces (different from the linear space of honestly-generated ciphertexts). In our proof, differently than in previous works, we need to carefully define the relationship between these different linear spaces. In particular, it is necessary to make sure that re-randomizations of the challenge ciphertexts still lie in the prescribed linear space (and thus can be identified by our technique when answering $\diamond$). More technically, a ciphertext for our scheme can be parsed as a vector $[\vec{x}]$ in the source group (the CPA-part of the ciphertext) plus two zero-knowledge proofs of consistency. The vector $[\vec{x}]$ for a well-formed ciphertext lies in the affine space defined by the encrypted message and the span of a matrix $[\vec{D}^*]$ which is part of the public key. Re-randomization works by summing up a random vector from the span of $\vec{D}^*$ to $\vec{x}$ (and updating the proofs accordingly). To apply the adaptive partitioning techniques, we move the challenge ciphertexts back and forth from two well-crafted distinct super spaces of $\vec{D}^*$. Thanks to this choice, we can recognize the challenge ciphertexts after re-randomization by multiplying the decrypted ciphertext by a matrix orthogonal[2] to $\vec{D}^*$. Thus, like previous

[2]This operation could be roughly interpreted as an "extended decryption" of the ciphertexts (since $\vec{D}^*$ encodes partial information of the secret key), however, we are not only interested to identify the encrypted

adaptive partitioning approaches, we separate the randomness space of the PKE scheme into an honest part (the span of $\vec{D}^*$) and a normally unused part (spanned by the vectors in the mentioned super spaces, independent of $\vec{D}^*$) that is also used to hide the messages. In our view, the main technical insight is that the span of $\vec{D}^*$ is used for re-randomization, while the other space is kept fixed for the challenge ciphertexts. We highlight that in order for the aforementioned strategy to work smoothly, we preferred to follow a flavor of adaptive partitioning as in Gay *et al.* [GHKP18], where secret keys are randomized, instead of the original strategy of Hofheinz [Hof17], where ciphertexts are randomized. Finally, the original adaptive partitioning strategy relies on the pairwise universality of a hash proof system [CS98] that guarantees simpler statements about linear languages. We adapt this proof system to re-randomizable statements by considering higher-dimensional languages and refining the "core lemma for Rand-RCCA" from [FFHR19]. We highlight that this lemma was designed for the single-ciphertext scenario, thus, some extra care is needed in our adaptive partitioning argument, more in detail, when defining the notion of *critical query*. In particular, a critical query is commonly defined as a decryption query for an ill-made ciphertext that would decrypt without errors under one of the randomized secret keys; the usual goal is to show that an adversary cannot make such a query. In our case, we need to refine this notion by additionally specifying when (allegedly) re-randomizations of challenge ciphertexts are critical. Since each one of the challenge ciphertexts is an ill-made ciphertext that decrypts correctly under one of the randomized keys, we cannot consider critical a re-randomization of such a challenge ciphertext when it decrypts correctly under the same randomized key. Thus, after having recognized a decryption query as a re-randomization we make sure that this ciphertext is decrypted only using a specific (a univocally linked) secret key; on the other hand, other kind of decryption queries can be safely decrypted with any of the secret keys. This rule allows eventually to use the lemma of [FFHR19], which provides security even given an interface for decryption of re-randomizations of one challenge ciphertext under one specific secret key.

7.3 Pair-wise independence of a projective hash function

The main technical tool employed by [FFHR19], to which they refer as their "core lemma", roughly speaking says that, for any $\vec{u} \in \mathbb{Z}_q^{d+1}$, the projective hash function with hash key $\vec{f}, \vec{F}$ that maps $\vec{v}$ to $(\vec{f} + \vec{F}\vec{v})^\top \vec{u}$ is pair-wise independent with respect to the quotient set $\mathbb{Z}_q^{d+2}/span(\vec{E})$ when given as side information the matrix $\vec{F}\vec{E}$ where $\vec{E} \in \mathbb{Z}_q^{d+2\times d}$. We generalize their result to $\vec{u} \in \mathbb{Z}_q^n$ and $\vec{E} \in \mathbb{Z}_q^{n'\times d}$ for any $n > d$ and $n' > d+1$. For the sake of clarity, in this chapter we prefer to call this lemma the "Rand-RCCA lemma", rather than "core lemma" (for Rand-RCCA) as in [FFHR19], because the core technical parts of our work and theirs are different.

Lemma 7.3.1 (Rand-RCCA Lemma)**.** *Let d be a positive integer. For any matrix $\vec{D} \in \mathbb{Z}_q^{n\times d}$,*

message but also to uniquely link the decrypted (possibly re-randomized) ciphertext with one of the challenge ciphertexts.

$\vec{E} \in \mathbb{Z}_q^{n' \times d}$ *where* $n > d$ *and* $n' > d+1$*, and any (possibly unbounded) adversary* A*:*

$$\Pr\left[\begin{array}{c} \vec{u} \notin span(\vec{D}) \\ (\vec{v} - \vec{v}^*) \notin span(\vec{E}) \\ z = (\vec{f} + \vec{F}\vec{v})^\top \vec{u} \end{array} : \begin{array}{c} \vec{f} \leftarrow\!\$\ \mathbb{Z}_q^n, \vec{F} \leftarrow\!\$\ \mathbb{Z}_q^{n \times n'}, \\ (z, \vec{u}, \vec{v}) \leftarrow\!\$\ \mathcal{A}^{\mathcal{O}\cdot}(\vec{D}, \vec{E}, \vec{f}^\top \vec{D}, \vec{F}^\top \vec{D}, \vec{F}\vec{E}) \end{array}\right] \leq \frac{n \cdot n'}{q}$$

where the adversary outputs a single query $\vec{v}^*$ *to* $\mathcal{O}$ *that returns* $\vec{f} + \vec{F} \cdot \vec{v}^*$*.*

Proof. The original lemma was proved in [FFHR19] for slightly different parameters, namely for matrices $\vec{D} \in \mathbb{Z}_q^{d+1 \times d}$ and $\vec{E} \in \mathbb{Z}_q^{d+2 \times d}$ and with upper bound $1/q$ to the winning probability of the adversary. We show here that their lemma holds for generic $n, n' > d$. To show this we reduce to the case proved in [FFHR19]. We parse the matrices $\vec{D}$ and $\vec{E}$ as follows:

$$\vec{D} = \begin{pmatrix} \vec{\bar{D}} \\ \vec{D}' \end{pmatrix} \cdot \vec{P} \text{ and } \vec{E} = \begin{pmatrix} \vec{E}' \\ \vec{\bar{E}} \end{pmatrix} \vec{P}'$$

where $\vec{\bar{D}}$ has $d+1$ rows and $\vec{\bar{E}}$ has $d+2$ rows and $\vec{P}, \vec{P}'$ are two uniformly random permutation matrices. Now consider the reduction to the original lemma where we use the matrices $\vec{\bar{D}}, \vec{\bar{E}}$, We obtain from the challenger the public parameters $(\vec{\bar{f}}^\top \vec{\bar{D}}, \vec{\bar{F}}^\top \vec{\bar{D}}, \vec{\bar{F}}\vec{\bar{E}})$ for uniformly random $\vec{\bar{f}}, \vec{\bar{F}}$. Now consider the vector $\vec{f}$ and matrix $\vec{F}$ implicitly defined as follows:

$$\vec{f} = \begin{pmatrix} \vec{\bar{f}} \\ \vec{f}' \end{pmatrix} \text{ and } \vec{F} = \begin{pmatrix} \vec{F}' & \vec{\bar{F}} \\ \vec{F}'' & \vec{F}''' \end{pmatrix}$$

where $\vec{f}', \vec{F}', \vec{F}'', \vec{F}'''$ are uniformly random, and can be sampled by the reduction. Notice that we can compute the public parameters for the adversary as follows:

$$\vec{f}^\top \vec{D} \leftarrow (\vec{\bar{f}}^\top \vec{\bar{D}} + \vec{f}'^\top \vec{D}')\vec{P}, \qquad \vec{F}^\top \vec{D} \leftarrow \begin{pmatrix} \vec{F}'^\top \vec{\bar{D}} + \vec{F}''^\top \vec{D}' \\ \vec{\bar{F}}^\top \vec{\bar{D}} + \vec{F}'''^\top \vec{D}' \end{pmatrix} \vec{P}$$

$$\vec{F}\vec{E} \leftarrow \begin{pmatrix} \vec{F}'\vec{E}' + \vec{\bar{F}}\vec{\bar{E}} \\ \vec{F}''\vec{E}' + \vec{F}'''\vec{\bar{E}} \end{pmatrix} \vec{P}'$$

moreover, we can handle the oracle query $\vec{v}^* \in \mathbb{Z}_q^{n'}$ of the adversary using the oracle of the reduction. Specifically, let $\vec{\bar{v}}^*$ be the last $d+2$ rows of $\vec{v}^*$ and $\vec{v}'$ be remaining rows, the reduction queries $\vec{\bar{v}}$ to its own oracle and obtain $\vec{\bar{y}} = \vec{\bar{f}} + \vec{\bar{F}}\vec{\bar{v}}^*$. The reduction can return to the adversary

$$\vec{y} \leftarrow \begin{pmatrix} \vec{\bar{y}} + \vec{F}'\vec{v}' \\ \vec{f}' + (\vec{F}''|\vec{F}''')\vec{v}^* \end{pmatrix}$$

Finally, the adversary returns $(z, \vec{u}, \vec{v})$, the reduction outputs tuple $(\bar{z}, \vec{\bar{u}}, \vec{\bar{v}})$ where $\vec{\bar{u}}$ are the first $d+1$ rows of $\vec{u}$, $\vec{\bar{v}}$ are the last $d+2$ rows of $\vec{v}$ and:

$$\bar{z} \leftarrow z - (\vec{F}'\vec{v}')^\top \vec{\bar{u}} - (\vec{f}' + \vec{F}''\vec{v}' + \vec{F}'''\vec{\bar{v}})^\top \vec{u}'$$

where $\vec{v}', \vec{u}'$ are the remaining rows of respectively $\vec{v}, \vec{u}$.

We need to prove that the forging probability of our reduction is as claimed in the statement of the lemma. To do so notice that if $\vec{u} \notin span(\vec{D})$ then we can write $\vec{u} = \vec{D}\vec{r} + \vec{z}$ for a non-zero vector $\vec{z}$. Thus, $\vec{\vec{u}} \notin span(\vec{\vec{D}})$ if there exists a non-zero entry in the first $d+2$ rows of $\vec{z} \cdot \vec{P}$. In the worst case $\vec{z}$ has only one non-zero coordinate. Thus, the probability that $\vec{\vec{u}} \notin span(\vec{\vec{D}})$ is $\frac{d+1}{n}$. A similar argument shows that the probability of $\vec{\vec{v}} - \vec{\vec{v}}^* \notin span(\vec{\vec{E}})$ is $\frac{d+2}{n'}$. Notice that the permutation matrices $\vec{P}, \vec{P}'$ are independent of the view of the adversary, thus the aforementioned events are independent of the winning probability of $\mathcal{A}$, which concludes the proof of the lemma. □

7.4 Non-Interactive Designated Verifier Proofs

In this section we define Non-Interactive Designated-Verifier Proof Systems (NIDVPS) with tags and their malleable counterparts.

The following definition are adapted from [Hof17].

Definition 7.4.1 (Proof system)**.** *Let $\mathcal{L} = \{\mathcal{L}_{pars}\}$ be a family of languages with $\mathcal{L}_{pars} \subseteq \mathcal{X}_{pars}$, and with efficiently computable witness relation $\mathcal{R}$. A non-interactive proof system (NIPS)* $\mathbf{PS} = (\mathsf{PGen}, \mathsf{PPrv}, \mathsf{PVer}, \mathsf{PSim})$ *for $\mathcal{L}$ consists of the following PPT algorithms:*

- $\mathsf{PGen}(1^\lambda, pars)$ *outputs a proving key ppk, a verification key psk.*
- $\mathsf{PPrv}(ppk, x, w)$*, $x \in \mathcal{L}$ and $\mathcal{R}(x, w) = 1$, outputs a proof π.*
- $\mathsf{PVer}(psk, x, \pi)$*, $x \in \mathcal{X}$ and a proof π, outputs a verdict $b \in \{0, 1\}$.*
- $\mathsf{PSim}(psk, x)$*, $x \in \mathcal{L}$, outputs a proof π.*

Completeness: *For all pars, all (ppk, psk) in the range of $\mathsf{PGen}(1^\lambda, pars)$, all $x \in \mathcal{L}$, and all w with $\mathcal{R}(pars, x, w) = 1$, we have $\mathsf{PVer}(psk, x, \mathsf{PPrv}(ppk, x, w)) = 1$.*

When $ppk \neq psk$ we say that the proof system is *designated verifier*. In the definition above we let the verification and proving key depend on the parameters of the relation, namely, the proof systems are *quasi-adaptive* as defined by Jutla and Roy [JR13]. All the NIPSs of this chapter are *structure-preserving*. The usual definition of structure-preserving postulates that all the public interfaces are vectors in the source groups, all the private material is in $\mathbb{Z}_q$ and all the algorithms can be described with pairing-product equations. We consider the version of the structure-preserving property of [FFHR19] where the proof π could lie in the target group.

7.4.0.1 Benign Proof Systems

All relevant security properties of a benign NIDVPS are condensed in the following definitions, taken verbatim from [Hof17].

Definition 7.4.2 (Benign proof system)**.** *Let* **PS** *be an NIDVPS for $\mathcal{L}$ as in Definition 7.4.1, and let $\mathcal{L}^{\text{sim}} = \{\mathcal{L}^{\text{sim}}_{pars}\}$, $\mathcal{L}^{\text{ver}} = \{\mathcal{L}^{\text{ver}}_{pars}\}$, and $\mathcal{L}^{\text{snd}} = \{\mathcal{L}^{\text{snd}}_{pars}\}$ be families of languages. We say that* **PS** *is* $(\mathcal{L}^{\text{sim}}, \mathcal{L}^{\text{ver}}, \mathcal{L}^{\text{snd}})$-benign *if the following properties hold:*

(Perfect) zero-knowledge. *For all pars, all* (ppk, psk) *that lie in the range of* $\mathsf{PGen}(1^\lambda, pars)$*, and all* $x \in \mathcal{L}$ *and* w *with* $\mathcal{R}(pars, x, w) = 1$*, we have that the distribution* $\mathsf{PPrv}(ppk, x, w)$ *is equivalent to* $\mathsf{PSim}(psk, x)$*.*

(Statistical) $(\mathcal{L}^{\text{sim}}, \mathcal{L}^{\text{ver}}, \mathcal{L}^{\text{snd}})$**-soundness.** *Let* $\mathbf{Exp}^{\mathtt{snd}}_{\mathcal{A},\mathbf{PS}}$ *be the game played by* $\mathcal{A}$ *in Fig. 7.1. Let* $\mathbf{Adv}^{\mathtt{snd}}_{\mathbf{PS},\mathcal{A}}(\lambda)$ *be the probability that* $\mathbf{Exp}^{\mathtt{snd}}_{\mathcal{A},\mathbf{PS}}(\lambda) = 1$*. We require that for all (possibly unbounded)* $\mathcal{A}$ *that only make a polynomial number of oracle queries,* $\mathbf{Adv}^{\mathtt{snd}}_{\mathbf{PS},\mathcal{A}}(\lambda)$ *is negligible.*

7.4.0.2 Non-Interactive Zero-Knowledge Proof Systems

We adapt Definition 7.4.1 for the case of publicly verifiable proof systems by requiring the prover key and the verification key to be identical, and we refer to such key as the *common reference string*. (Nontrivial) proof systems with this syntax are commonly called zero-knowledge proof systems (NIZKs).

Notice that in the syntax of proof system we give in Definition 7.4.2 both the simulator PSim and the verifier PVer receive as input the verification key, while in the usual definition of NIZK the simulator receives a simulation trapdoor. This difference is only syntactical.

We say that a NIZK $\mathbf{PS}$ for $\mathcal{L}$ is *adaptively sound* if it is statistically $(\emptyset, \mathcal{L}, \emptyset)$-sound according to Definition 7.4.2.

Definition 7.4.3. *Let* $\mathbf{PS}$ *be a NIPS for* $\mathcal{L}$ *as in Definition 7.4.1, we say that* $\mathbf{PS}$ *is* (ϵ, T)-composable zero-knowledge *if there exists a PPT algorithm* $\overline{\mathsf{PGen}}$ *such that:*

- *For all pars, the distributions induced by the first output of* $\mathsf{PGen}(1^\lambda, pars)$ *and* $\overline{\mathsf{PGen}}(1^\lambda, pars)$ *are* ϵ*-close for any adversary with running time* T*.*
- *For all pars, all* (ppk, psk) *that lie in the range of* $\overline{\mathsf{PGen}}(1^\lambda, pars)$*, and all* $x \in \mathcal{L}$ *and* w *with* $\mathcal{R}(pars, x, w) = 1$*, we have the following equivalence of distributions:*

$$\mathsf{PPrv}(ppk, x, w) \quad \equiv \quad \mathsf{PSim}(psk, x).$$

7.4.0.3 Malleable NIPS

We use the definitional framework of Chase *et al.* [CKLM12] for malleable proof systems. For simplicity of the exposition we consider only the unary case for transformations (see the aforementioned paper for more details). Moreover, we adapt their definition to the quasi-adaptive setting by having transformation that depends on the *pars*. Let $T = (T_{\mathsf{el}}, T_{\mathsf{wit}})$ be a pair of efficiently computable functions, that we refer to as a *transformation*.

Definition 7.4.4 (Admissible transformation)**.** *An efficient relation* $\mathcal{R}$ *is* closed *under a transformation* $T = (T_{\mathsf{el}}, T_{\mathsf{wit}})$ *if for any* $(pars, x, w) \in \mathcal{R}$ *the pair* $(pars, T_{\mathsf{el}}(pars, x), T_{\mathsf{wit}}(w)) \in \mathcal{R}$*. If* $\mathcal{R}$ *is closed under* T *then we say that* T *is* admissible *for* $\mathcal{R}$*. Let* $\mathcal{T}$ *be a set of transformations, if for every* $T \in \mathcal{T}$*,* T *is admissible for* $\mathcal{R}$*, then* $\mathcal{T}$ *is an* allowable *set of transformations.*

We are ready to define malleable proof systems.

Experiment $\mathbf{Exp}^{\mathtt{snd}}_{\mathcal{A},\mathbf{PS}}$

$b \leftarrow 0$
$pars \leftarrow \mathcal{A}(1^\lambda)$
$(ppk, psk) \leftarrow\!\!\$\ \mathsf{PGen}(1^\lambda, pars)$
$\mathcal{A}^{\mathcal{O}_{\mathbf{sim}}(\cdot),\mathcal{O}_{\mathbf{ver}}(\cdot,\cdot)}(ppk)$
return b

Oracle $\mathcal{O}_{\mathbf{ver}}(x, \pi)$

if $x \in \mathcal{L}^{\mathrm{ver}}_{pars}$:
  return $\mathsf{PVer}(psk, x, \pi)$
if $x \in \mathcal{X}_{pars} \setminus \mathcal{L}^{\mathrm{snd}}_{pars} \wedge \mathsf{PVer}(psk, x, \pi) \stackrel{?}{=} 1$:
  $b \leftarrow 1$
return $\bot$

Oracle $\mathcal{O}_{\mathbf{sim}}(x)$

if $x \in \mathcal{L}^{\mathrm{sim}}_{pars}$:
  return $\mathsf{PSim}(psk, x)$
else return $\bot$

Experiment $\mathbf{Exp}^{\mathtt{der\text{-}priv}}_{\mathcal{A},\mathbf{PS}}$

$b^* \leftarrow\!\!\$\ \{0,1\}$
$(ppk, psk) \leftarrow\!\!\$\ \mathsf{PGen}(1^\lambda, pars)$
$(x, w, \pi, T) \leftarrow \mathcal{A}(ppk, psk)$
if $\mathsf{V}(ppk, x, \pi) \stackrel{?}{=} 0 \vee R(x, w) \stackrel{?}{=} 0$:
  $b \leftarrow\!\!\$\ \{0,1\}$
  return b
if $b^* \stackrel{?}{=} 0$:
  $\pi' \leftarrow \mathsf{PPrv}(ppk, T_{\mathsf{el}}(pars, x), T_{\mathsf{wit}}(w))$
else $\pi' \leftarrow \mathsf{PEvl}(ppk, x, \pi, T)$
$b \leftarrow \mathcal{A}(\pi')$
return $b \stackrel{?}{=} b^*$

Figure 7.1: Security experiments for benign soundness and derivation privacy of NIPS.

Definition 7.4.5 (Malleable NIPS)**.** *Let* **PS** *be an NIPS for* $\mathcal{L}$ *as in Definition 7.4.1, and let* $\mathsf{PEvl}(ppk, x, \pi, T)$ *be a PPT algorithm that takes as inputs ppk, an instance* x*, a proof* π*, and a transformation* $T \in \mathcal{T}$*, and it outputs a proof* π'*. We say that* **PS** *and* PEvl *form a* malleable proof system *for* $\mathcal{L}$ *with set* $\mathcal{T}$ *of allowable transformations for* $\mathcal{R}$*, if, for all pars,* (ppk, psk) *that lie in the range of* $\mathsf{PGen}(1^\lambda, pars)$*, all* $T \in \mathcal{T}$*, and all* x, π *we have* $\mathsf{PVer}(psk, T_{\mathsf{el}}(pars, x), \pi') = 1$ *if and only if* $\mathsf{PVer}(psk, x, \pi) = 1$*.*

Definition 7.4.6 (Derivation Privacy)**.** *Let* **PS** *be a malleable NIPS for* $\mathcal{L}$ *with relation* $\mathcal{R}$ *and an allowable set of transformations* $\mathcal{T}$ *and corresponding* PEvl*. We say that* **PS** *is* derivation private *if for any PPT adversary* $\mathcal{A}$*:*

$$\mathbf{Adv}^{\mathtt{der\text{-}priv}}_{\mathcal{A},\mathbf{PS}}(\lambda) := \left|\Pr\left[\mathbf{Exp}^{\mathtt{der\text{-}priv}}_{\mathcal{A},\mathbf{PS}}(\lambda) = 1\right] - \tfrac{1}{2}\right| \in \mathsf{negl}(\lambda)$$

where $\mathbf{Exp}^{\mathtt{der\text{-}priv}}$ *is the game described in Fig. 7.1. Moreover, we say that* **PS** *is* perfectly *(resp.* statistically*)* derivation private *when for any (possibly unbounded) adversary the advantage above is 0 (resp. negligible).*

Similarly to [FFHR19], we also require a technical property to show re-randomizability of our encryption scheme that we call *tightness for proofs*, which roughly speaking says that it is hard to find a proof for a valid instance that does not lie in the set of the proofs created by the prover.

Definition 7.4.7 (Tightness for Proofs)**.** *We say that a* NIZK *has* tight proofs *if for any (possibly unbounded) adversary $\mathcal{A}$ the following probability is negligible in the security parameter:*

$$\Pr\left[\begin{array}{c} \pi \notin \{\mathsf{PPrv}(ppk, x, w; \omega)\}_{\omega \in \{0,1\}^\lambda} \\ \wedge \ \ \mathsf{PVer}(psk, x, \pi) = 1 \end{array} : \begin{array}{c} (ppk, psk) \leftarrow\!\!\$ \ \ \mathsf{PGen}(1^\lambda, pars) \\ (x, w, \pi) \leftarrow \ \ \mathcal{A}(ppk) \end{array}\right]$$

We notice that the property above is true for Groth-Sahai Proofs, for the quasi-adaptive proof system of Kiltz and Wee [KW15] and for designated-verifier proof system based on projective hash functions. In particular, for GS proofs, for any commitment to the witness, the prover generates a proof that is uniformly distributed over the set of all the possible valid proofs. The proofs of Kiltz and Wee and the proofs for proof system based on projective hash functions are unique, therefore the condition is trivially true.

7.4.1 Our Malleable NIDVPS based on type-1 pairing

The scheme we propose is inspired by the work of [ABP15] which shows how to instantiate the disjunction of two SPHFs for two languages based on diverse vector spaces. We do not need such a functionality for our benign proof system, indeed our proof system is for linear space, i.e. the prover can generate proofs for elements that belong to the span of matrix $\vec{D}$. On the other hand, the security property of our benign proof system allows for soundness even in presence of simulated proofs for elements in two (possibly distinct) super-spaces of the prescribed linear space. In other words, while [ABP15] enables for SPHFs for (arbitrary) disjoint linear spaces, our goal is to enable for proof system for linear spaces with enhanced soundness properties (w.r.t. simulated proofs from disjoint super-spaces).

Construction 7.4.1. *Let $\vec{D} \in \mathbb{Z}_q^{n \times d}$.*

- $\mathsf{PGen}(pars)$ *parses pars as* $\mathtt{prm}_G, [\vec{D}]_1 \in \mathbb{G}_1^{n \times d}$ *where $n, d \in \mathbb{N}$, samples $\vec{k} \leftarrow\!\!\$ \ \mathbb{Z}_q^{n^2}$, let $\vec{I}_n$ be the identity matrix of dimension n, set:*

$$psk \leftarrow \vec{k} \text{ and } ppk \leftarrow (\vec{k}^\top [\vec{D} \otimes \vec{I}_n]_1, \vec{k}^\top [\vec{I}_n \otimes \vec{D}]_1, \vec{k}^\top [\vec{D} \otimes \vec{D}]_T)$$

- $\mathsf{PPrv}(ppk, [\vec{u}]_1, \vec{r})$ *computes* $\pi \leftarrow \vec{k}^\top [\vec{D} \otimes \vec{D}]_T \cdot (\vec{r} \otimes \vec{r})$ *for* $[\vec{u}]_1 = [\vec{D}]_1 \vec{r}$
- $\mathsf{PSim}(psk, [\vec{u}]_1)$ *computes* $\pi \leftarrow \vec{k}^\top ([\vec{u}]_1 \otimes [\vec{u}]_1)$
- $\mathsf{PVer}(psk, [\vec{u}]_1, \pi)$ *returns* 1 *if and only if* $\vec{k}^\top ([\vec{u}]_1 \otimes [\vec{u}]_1) \stackrel{?}{=} \pi$

We show that the following **PS** is a NIPS for $\mathcal{L} = span([\vec{D}]_1)$. The first two vectors in the ppk are necessary to enable for the malleability of the proof system. While the third element of the public key could be efficiently derived from the previous two, we decide to publish it to speed up re-randomization and proving time. Consider the set $\mathcal{T}$ of admissible transformations for $\mathbb{Z}_q^n$:

$$\mathcal{T} = \{T : T_{\mathsf{el}}(pars, [\vec{u}]_1) = [\vec{u}]_1 + [\vec{D}]_1 \hat{\vec{r}};\ T_{\mathsf{wit}}(\vec{r}) = \vec{r} + \hat{\vec{r}}\} \tag{7.1}$$

We note that any transformation T in the set above is uniquely determined by the vector $\vec{\hat{r}}$, thus, whenever it is clear from the context, we will simply use $\vec{\hat{r}}$ to identify the transformation. Let $\mathsf{PEvl}(ppk, \vec{\hat{r}}, [\vec{u}]_1, \pi)$ the algorithm that computes

$$\hat{\pi} \leftarrow \pi + \vec{k}^\top [\vec{I}_n \otimes \vec{D}]_1 \cdot [\vec{u} \otimes \vec{\hat{r}}]_1 + \vec{k}^\top [\vec{D} \otimes \vec{I}_n]_1 \cdot [\vec{\hat{r}} \otimes \vec{u}]_1 + \vec{k}^\top [\vec{D} \otimes \vec{D}]_T \cdot \vec{\hat{r}} \otimes \vec{\hat{r}}.$$

We show that **PS** and PEvl form a malleable proof system for the set of transformation $\mathcal{T}$ and the language $\mathcal{L}$.

Theorem 7.4.1. *Let $\mathcal{L} = span([\vec{D}]_1)$ and let $\mathcal{L}^{\text{snd}} = \mathcal{L}^{\text{sim}} = \{[\vec{u}]_1 : [\vec{u}]_1 = [\vec{D}_0]_1 \vec{r} \vee [\vec{u}]_1 = [\vec{D}_1]_1 \vec{r}\}$, and $\mathcal{L}^{\text{ver}} = \mathbb{Z}_q^n$, where $\vec{D}_i = \vec{D} \| \vec{\bar{D}}_i$ for $i \in \{0,1\}$, $\vec{D} \in \mathbb{Z}_q^{n \times d}$ and $\vec{\bar{D}}_0, \vec{\bar{D}}_1 \in \mathbb{Z}_q^{n \times d'}$. **PS** is a $(\mathcal{L}^{\text{sim}}, \mathcal{L}^{\text{ver}}, \mathcal{L}^{\text{snd}})$-benign proof system for $\mathcal{L}$ as long as $n^2 > 2n \cdot d + 2d'^2$, moreover, **PS** and PEvl form a malleable proof system for $\mathcal{L}$ and the set of transformation $\mathcal{T}$ defined in Eq. (7.1).*

Proof. In what follows, we prove each of the properties.

Completeness and Malleability. It is easy to prove that our benign proof system is complete, as by Eq. (2.1) for any $\vec{u} = \vec{D}\vec{r}$ we have $(\vec{u} \otimes \vec{u}) = (\vec{D} \otimes \vec{D}) \cdot (\vec{r} \otimes \vec{r})$. We prove that our scheme is *malleable* (Definition 7.4.5) with respect to set of transformation $\mathcal{T}$ defined in Eq. (7.1), i.e., we prove that for any $[\vec{u}]$ and any transformation $\vec{\hat{r}}$, a proof π for $[\vec{u}]$ verifies if and only if the proof $\hat{\pi}$ obtained executing PEvl on π and the transformation $\vec{\hat{r}}$ verifies for $[\vec{u} + \vec{D}\vec{\hat{r}}]$. For the first direction of the implication:

$$\begin{aligned}
\hat{\pi} &= \pi + \vec{k}^\top \left(\vec{I}_n \otimes \vec{D}\right) \cdot \left(\vec{u} \otimes \vec{\hat{r}}\right) + \vec{k}^\top \left(\vec{D} \otimes \vec{I}_n\right) \cdot \left(\vec{\hat{r}} \otimes \vec{u}\right) + \vec{k}^\top \left(\vec{D} \otimes \vec{D}\right) \cdot \left(\vec{\hat{r}} \otimes \vec{\hat{r}}\right) \\
&= \vec{k}^\top (\vec{u} \otimes \vec{u}) + \vec{k}^\top \left((\vec{I}_n \vec{u}) \otimes (\vec{D}\vec{\hat{r}})\right) + \vec{k}^\top \left((\vec{D}\vec{\hat{r}}) \otimes (\vec{I}_n \vec{u})\right) + \vec{k}^\top \left((\vec{D}\vec{\hat{r}}) \otimes (\vec{D}\vec{\hat{r}})\right) \\
&= \vec{k}^\top (\vec{u} \otimes \vec{u} + \vec{u} \otimes (\vec{D}\vec{\hat{r}}) + (\vec{D}\vec{\hat{r}}) \otimes \vec{u} + (\vec{D}\vec{\hat{r}}) \otimes (\vec{D}\vec{\hat{r}})) \\
&= \vec{k}^\top \left((\vec{u} + \vec{D}\vec{\hat{r}}) \otimes (\vec{u} + \vec{D}\vec{\hat{r}})\right)
\end{aligned}$$

We highlight that the second equation holds because of the definition of π and (2.1), while the third equation is obtained by grouping the previous line by $\vec{k}^\top$. The sequence of equations above also proves the other direction; indeed, if $\pi \neq \vec{k}^\top \vec{u} \otimes \vec{u}$, then $\hat{\pi} \neq \vec{k}^\top (\vec{u} + \vec{D}\vec{\hat{r}}) \otimes (\vec{u} + \vec{D}\vec{\hat{r}})$.

Soundness. We recall that $\vec{D} \in \mathbb{Z}_q^{n \times d}, \vec{\bar{D}}_i \in \mathbb{Z}_q^{n \times d'}$. If we only consider the view of the adversary given the verification key and the outputs of the simulation oracle we have that the proving key is uniformly distributed over a set of cardinality $q^{n^2 - 2nd - 2d'^2}$. Therefore, we require that $n^2 > 2n \cdot d + 2d'^2$ holds.

To see this, think of $\vec{k}$ as formal variable and notice that publishing the information $\vec{k}^\top \left(\vec{D} \otimes \vec{I}_n\right)$ counts for $n \cdot d$ equations; also, $\vec{k}^\top \left(\vec{I}_n \otimes \vec{D}\right)$ counts for $n \cdot d$ equations which in total gives us $2n \cdot d$ equations. Moreover, in order to simulate proofs for $[\vec{u}]_1 \in span([\vec{D}_i])$ the oracle gives away, at the worst case, the equations $\vec{k}^\top \left(\vec{\bar{D}}_i \otimes \vec{\bar{D}}_i\right)$ which count for d'^2 equations for each $i \in \{0,1\}$ which sum up to $2d'^2$ equations in total. Indeed, expanding $\vec{k}^\top \left(\vec{D}_i \otimes \vec{D}_i\right)$ we obtain $\vec{k}^\top \left(\vec{D} \otimes \vec{D} | \vec{\bar{D}}_i \otimes \vec{D} | \vec{D} \otimes \vec{\bar{D}}_i | \vec{\bar{D}}_i \otimes \vec{\bar{D}}_i\right)$. The vectors $\vec{k}^\top \left(\vec{\bar{D}}_i \otimes \vec{D}\right)$ and $\vec{k}^\top \left(\vec{D} \otimes \vec{\bar{D}}_i\right)$ can be computed given the proving key and $\vec{D}_0, \vec{D}_1$. In fact, computing $\vec{k}^\top \left(\vec{D} \otimes \vec{I}\right) \left(\vec{I} \otimes \vec{\bar{D}}_i\right)$

Experiment $\mathbf{Exp}^{\mathtt{mRCCA}}_{\mathcal{A},\mathsf{PKE}}(\lambda)$

$b^* \leftarrow\!\$\ \{0,1\}$
$\mathtt{prm} \leftarrow \mathsf{Setup}(1^\lambda)$
$b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathbf{kgen}}(),\mathcal{O}_{\mathbf{enc}}(\cdot,\cdot,\cdot),\mathcal{O}_{\mathbf{dec}}(\cdot)}(\mathtt{prm})$
return $b' \stackrel{?}{=} b^*$

Oracle $\mathcal{O}_{\mathbf{kgen}}()$

$z \leftarrow z+1$
$Q_z \leftarrow \mathrm{DisjointSet}()$
$(\mathsf{pk}_z, \mathsf{sk}_z) \leftarrow \mathsf{KGen}(\mathtt{prm})$
return pk_z

Oracle $\mathcal{O}_{\mathbf{enc}}(j, \mathtt{msg}_0, \mathtt{msg}_1)$

if $j \notin [z]$:
  return $\perp$
$\mathcal{Q}_j.\mathrm{union}(\{\mathtt{msg}_0, \mathtt{msg}_1\})$
$\mathtt{C} \leftarrow\!\$\ \mathsf{Enc}(\mathsf{pk}_j, \mathtt{msg}_{b^*})$
return $\mathtt{C}$

Oracle $\mathcal{O}_{\mathbf{dec}}(j, \mathtt{C})$

$\mathtt{msg} \leftarrow \mathsf{Dec}(\mathsf{sk}_j, \mathtt{C})$
$\mathcal{J} \leftarrow \mathcal{Q}_j.\mathrm{find}(\mathtt{msg})$
if $\mathcal{J} \neq \perp$:
  return $\diamond_{\mathcal{J}}$
return $\mathtt{msg}$

Figure 7.2: The multi-user and multi-ciphertext RCCA Security Experiment.

we obtain $\vec{k}^\top \left(\vec{D}I \otimes \vec{I}\vec{\bar{D}}_i\right) = \vec{k}^\top \left(\vec{D} \otimes \vec{\bar{D}}_i\right)$. And similarly, we can compute $\vec{k}^\top \left(\vec{\bar{D}}_i \otimes \vec{D}\right)$. In total, we are giving up $2n \cdot d + 2d'^2$ equations and the length of our key k is n^2.

Notice that the adversary can gather additional information about the proving key $\vec{k}$ through the verification oracle. Indeed, whenever it sends a query $([\vec{u}]_1, \pi)$ with $[\vec{u}]_1 \in \mathcal{L}^{\mathrm{ver}} \setminus \mathcal{L}^{\mathrm{snd}}$ either it wins the security game or the adversary learns that $\pi \neq \vec{k}^\top [\vec{u}]_1 \otimes [\vec{u}]_1$.

Consider the hybrid experiment $\mathbf{H}_j$ where the first j-th queries $([\vec{u}]_1, \pi)$ to the verification oracle with $[\vec{u}]_1 \notin \mathcal{L}^{\mathrm{snd}}$ are answered with 0, in particular, the bit b is left unmodified, while the remaining queries are handled as in the soundness experiment. Clearly, $\mathbf{H}_0$ is the original experiment, while $\mathbf{H}_Q$ where Q is an upper bound on the number of verification oracle queries made by the adversary is a trivial experiment where the adversary cannot win (since the bit b will never be set to 1), thus $\Pr[\mathbf{H}_Q = 1] = 0$. The distinguishing event between two consecutive hybrids is the event that the adversary wins the soundness experiment at the j-th query, which happens with probability $1/q^{n^2-2nd+2d'^2} \leq 1/q$, as it is the same as the event of guessing a uniformly random vector from a subspace of dimension $n^2 - 2nd + 2d'^2$ of $\mathbb{Z}_q^{n^2}$, thus $\Pr[\mathbf{H}_j = 1] \leq \Pr[\mathbf{H}_{j+1} = 1] + 1/q$. Finally, by the triangular equation and noticing that Q is polynomial in the security parameter we can conclude our proof of soundness.

Derivation Privacy and Zero-Knowledge. The scheme satisfies perfectly derivation privacy and zero-knowledge. For the former, notice that, for any $\vec{\hat{r}}$, we have that $\mathsf{PPrv}(ppk, [\vec{u}+\vec{D}\vec{\hat{r}}]_1, \vec{r}+\vec{\hat{r}}) = \vec{k}^\top [\vec{D} \otimes \vec{D}]_T \cdot ((\vec{r}+\vec{\hat{r}}) \otimes (\vec{r}+\vec{\hat{r}})) = \mathsf{PEvl}(ppk, \pi, \vec{\hat{r}})$. For the latter, given an instance $[\vec{u}]_1$ such that $[\vec{u}]_1 = [\vec{D}]_1\vec{r}$, we have that $\mathsf{PSim}(psk, [\vec{u}]_1) = \vec{k}^\top([\vec{u}]_1 \otimes [\vec{u}]_1) = \vec{k}^\top([\vec{D}\vec{r}]_1 \otimes [\vec{D}\vec{r}]_1) = \mathsf{PPrv}(ppk, [\vec{u}]_1, \vec{r})$. □

7.5 Rand RCCA PKE for multi-users and multi-ciphertexts

Definition 7.5.1 (multi-user and multi-ciphertext Replayable CCA Security)**.** *Consider the experiment* $\mathbf{Exp}^{\mathtt{mRCCA}}$ *in Fig. 7.2, with parameters* λ*, an adversary* $\mathcal{A}$*, and a PKE scheme* PKE*. We say that* PKE *is* indistinguishable secure under replayable chosen-ciphertext attacks in the multi-user and multi-ciphertext setting *(mRCCA-secure) if for any PPT adversary* $\mathcal{A}$*:*

$$\mathbf{Adv}^{\mathtt{mRCCA}}_{\mathcal{A},\mathsf{PKE}}(\lambda) := \left|\Pr\left[\mathbf{Exp}^{\mathtt{mRCCA}}_{\mathcal{A},\mathsf{PKE}}(\lambda) = 1\right] - \frac{1}{2}\right| \in \mathsf{negl}(\lambda).$$

In Fig. 7.2, for each user j we define $\mathcal{Q}_j$ to be a partition of the set of the challenge messages sent to the encryption oracle for the user j. To do so we use the classical *Disjoint-Set* (also called *Union-Find*) data structure from Tarjan [Tar75]. Whenever two challenge messages are submitted to the encryption oracle, indeed, we merge the sets to which they belong so that a future call to the guarded decryption oracle behaves consistently. This allows us to express in Fig. 7.2 the syntax of the encryption and the guarded decryption oracle in terms of three operations: DisjointSet() that allows to initialize the partition (initially empty), union(S) that adds to the partition the minimal subset of the challenge messages that contains the messages in S meanwhile maintaining invariant the partition property (i.e. a collection of disjoint sets), and find($\mathtt{msg}$) that returns the set in the partition where $\mathtt{msg}$ belongs to, or $\perp$ if $\mathtt{msg}$ is not in the set of challenge messages of the user j.

We confirm that our definition is indeed the right multi-user and multi-ciphertext extension of the IND-RCCA definition of [CKN03] by showing that our definition tightly implies the UC-RCCA definition of the same paper[3]. In Fig. 7.3 we recall the definition of the ideal functionality $\mathcal{F}_{\mathsf{RPKE}}$ which formalizes the notion of replay security for public-key encryption scheme in the universal composability model.

Theorem 7.5.1. *Let* PKE *be a PKE scheme with message space* $\mathcal{D}$*. There exists a simulator* $\mathcal{S}$ *such that for any static-corruption environment* $\mathcal{Z}$ *with running time* $T_{\mathcal{Z}}$ *there exists an adversary* $\mathcal{B}$ *whose running time is* $O(T_{\mathcal{Z}}(\lambda))$ *such that:*

$$\left|\Pr[REAL_{\mathcal{Z},\Pi_{\mathsf{PKE}}}(\lambda) = 1] - \Pr\left[IDEAL^{\mathcal{F}_{\mathsf{RPKE}}}_{\mathcal{Z},\mathcal{S}}(\lambda) = 1\right]\right| \leq 2\mathbf{Adv}^{\mathtt{mumc-RCCA}}_{\mathcal{B},\mathsf{PKE}}(\lambda) + \tfrac{T_{\mathcal{Z}}}{|\mathcal{D}|}$$

Proof. The simulator is the same as the one presented at page 14 of [CKN03]. In particular:

- At first activation of the simulator it sets the set of random messages $\mathcal{M}^*$ to be the empty set.
- When $\mathcal{S}$ receives the message $(\mathtt{KeyGen}, id)$ from the ideal functionality, then it runs $\mathsf{pk}, \mathsf{sk} \leftarrow\!\!\$\ \mathsf{KGen}(\mathtt{prm})$, returns pk to the ideal functionality and stores the key-pair.
- When $\mathcal{S}$ receives a message $(\mathtt{Encrypt}, id, e', P_j)$ if $e' = \mathsf{pk}$ then the simulator samples a uniformly random message $\mathtt{msg}^*$ from the set $D_\lambda \setminus \mathcal{M}^*$ and adds $\mathtt{msg}^*$ into the set $\mathcal{M}^*$, computes $\mathtt{C} \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{pk}, \mathtt{msg}^*)$, store te tuple $(\mathtt{C}, \mathtt{msg}^*)$ and returns $\mathtt{C}$, otherwise it additionally receives a message $\mathtt{msg}$ from the ideal functionality, it returns $\mathsf{Enc}(e', \mathtt{msg})$.

[3] In [CKN03], the IND-RCCA notion implies the UC-RCCA notion with a loss of security that is proportional to the running time of the environment.

Key Generation: Upon receiving a value $(\texttt{KeyGen}, id)$ from some party P_i

1. Hand $(\texttt{KeyGen}, id)$ to the adversary.
2. Receive a value e from the adversary, and hand e to P_i.
3. If this is the first activation then record the value e.

Encryption: Upon receiving from some party P_j a value $(\texttt{Encrypt}, id, e', m)$

1. If $m \notin D_k$ then return an error message to P_j.
2. If $m \in D_k$ then hand $(\texttt{Encrypt}, id, e', P_j)$ to the adversary. (If $e' \neq e$ or e is not yet defined then hand also the entire value m to the adversary.)
3. Receive a ciphertext c from the adversary, record the pair (c, m), and hand c to P_j. If $e' \neq e$ or e is not yet defined then do not record the pair (c, m). If the tag c already appears in a previously recorded pair then return an error message to P_j.

Decryption: Upon receiving a value $(\texttt{Decrypt}, id, c)$ from P_i (and P_i only)

1. If there is a recorded pair (c, m) then hand m to P_i.
2. Otherwise, hand the value $(\texttt{Decrypt}, id, c)$ to the adversary, and receive a value (α, v) from the adversary. If $\alpha = \texttt{plaintext}$ then hand v to P_i. If $\alpha = \texttt{ciphertext}$ then find a stored pair (c', m) such that $c' = v$, and hand m to P_i. (If no such c' is found then halt.)

Figure 7.3: The ideal functionality $\mathcal{F}_{\mathsf{RPKE}}$, when parameterized by message domain ensemble $D = \{D_\lambda\}_{\lambda \in \mathbb{N}}$ and security parameter λ, and interacting with an adversary S, and parties $P_1, \ldots, P_n$.

- When $\mathcal{S}$ receives a message $(\texttt{Decrypt}, id, c)$ from the ideal functionality it first computes $\texttt{msg} = \mathsf{Dec}(\mathsf{sk}, c)$. If it exists a recorded tuple $(\texttt{C}, \texttt{msg})$ then it returns $(\texttt{ciphertext}, \texttt{C})$ otherwise it returns $(\texttt{plaintext}, \texttt{msg})$.

We reduce to our multi-ciphertext RCCA security notion of Definition 7.5.1. Consider the reduction $\mathcal{B}$ that runs the environments $\mathcal{Z}$ and answers its messages as follows:

- Upon message $(\texttt{KeyGen}, id)$ to party P_i, if this is the first activation then $\mathcal{B}$ queries its own key generation oracle and receives pk which returns as output of P_i. Else it generates a key-pair and returns pk as output of P_i.
- Upon message $(\texttt{Encrypt}, id, e', \texttt{msg})$ from $\mathcal{Z}$ to a party P_j, it checks that $\texttt{msg} \in D_\lambda$, and otherwise it returns an error as output of P_j. If $e = \mathsf{pk}$ it samples a random message $\texttt{msg}^* \leftarrow\!\!\$\ D_\lambda \backslash \mathcal{M}^*$ and stores the message in the list $\mathcal{M}^*$ of random messages. Additionally,

it asserts that the message from the environment $\mathtt{msg}$ is not present in $\mathcal{M}^*$, if such event happens it **abort**. It sets $(\mathtt{msg}_0, \mathtt{msg}_1) \leftarrow (\mathtt{msg}, \mathtt{msg}^*)$ and queries its own encryption oracle obtaining $\mathtt{C}$ which returns as output of P_j.

- Upon message $(\mathtt{Decrypt}, id, c)$ sent by the environment to P_i, it queries with c the decryption oracle. If the decryption oracle return $\diamond_J$ let $A = J \setminus \mathcal{M}^*$, if $|A| > 1$ then the reduction **aborts**, else $A = \{\mathtt{msg}\}$, and it returns $\mathtt{msg}$ as output of P_j. If the decryption oracle returns $\mathtt{msg}$ then it returns $\mathtt{msg}$ as output of P_j.

Eventually, the reduction outputs the same decision bit of the environment $\mathcal{Z}$.

It is easy to check that if we condition on the event that the reduction $\mathcal{B}$ does not abort then the view of the environment is exactly the same as in the real world if the challenge bit of the RCCA experiment is 0 and exactly the same as in the ideal world when the challenge bit of the RCCA experiment is 1.

Let **Abort** be such event, we compute the probability of **Abort** conditioned on the challenge bit being 0. Notice that, since the event can be checked efficiently, there exists a reduction $\mathcal{B}'$ with running time less or equal to the running time o $\mathcal{B}$ such that $\Pr[\mathbf{Abort}|b^* = 1] \leq \Pr[\mathbf{Abort}|b^* = 1] + \mathbf{Adv}^{\mathtt{mRCCA}}_{\mathcal{B}',\mathsf{PKE}}(\lambda)$. The reduction $\mathcal{B}'$ runs $\mathcal{B}$ and returns 1 if $\mathcal{B}$ aborts, and otherwise it outputs a random bit.

Notice that when $b = 0$ the random messages in $\mathcal{M}^*$ are not in the view of $\mathcal{Z}$ (since the ciphertexts contain encryption of the real messages), thus the probability that **Abort** happens because the environment sends $(\mathtt{Encrypt}, id, e', \mathtt{msg})$, where $\mathtt{msg} \in \mathcal{M}^*$, is upper-bounded by $|\mathcal{M}^*|/|D_\lambda|$. Moreover, notice that when the decryption oracle returns $\diamond_J$, since the random messages $\mathtt{msg}^*$ chosen by the reduction are all different, the common element in all the encryption queries that made them to be included in J, by the Definition 7.5.1, must be the real message sent by the $\mathcal{Z}$.

Summing up all together, taking the maximum between the advantage of $\mathcal{B}$ and $\mathcal{B}'$, and noticing that $|\mathcal{M}^*| \leq T_{\mathcal{Z}}$ we obtain the bound in the statement of the theorem. □

7.6 Our Rand-RCCA PKE Scheme

We present our scheme in Fig. 7.4. With the goal of improving readability for developers, all the operations (and in particular the pairing operations) in the figure are described explicitly using e for the pairing and $\cdot$ for the exponentiations. The scheme can be summarized as a type-1 pairing group version of the scheme in [FFHR19] where we additionally append a benign proof to prove almost tight-security.

The main technical component from [FFHR19] to obtain RCCA security is the *consistency check* at decryption time which checks that

$$[y]_T \stackrel{?}{=} \vec{f}^\top [\vec{u}]_T + [\vec{x}]_1^\top \vec{F}^\top [\vec{u}]_1$$

Perfect Re-randomizability. We prove that the scheme presented in Fig. 7.4 is perfectly re-randomizable, according to Definition 2.4.1. We start by introducing the following lemma.

Lemma 7.6.1. *For any $[\vec{x}]_1$ and $\vec{r}$, let $[\vec{\hat{x}}]_1 = [\vec{x}]_1 + [\vec{D}^*]_1 \vec{\hat{r}}$, we have that*

$$\begin{aligned}(\vec{f}^\top + [\vec{\hat{x}}]_1^\top \vec{F}^\top)[\vec{\hat{u}}]_1 = (\vec{f}^\top + [\vec{x}]_1^\top \vec{F}^\top)[\vec{u}]_1 + [\vec{f}^\top \vec{D}]_T \cdot \vec{\hat{r}} \\ + e([\vec{x}]_1, [\vec{F}^\top \vec{D}]_1 \cdot \vec{\hat{r}}) + e([\vec{F}\vec{D}^*]_1 \cdot \vec{\hat{r}}, [\vec{\hat{u}}]_1)\end{aligned}$$

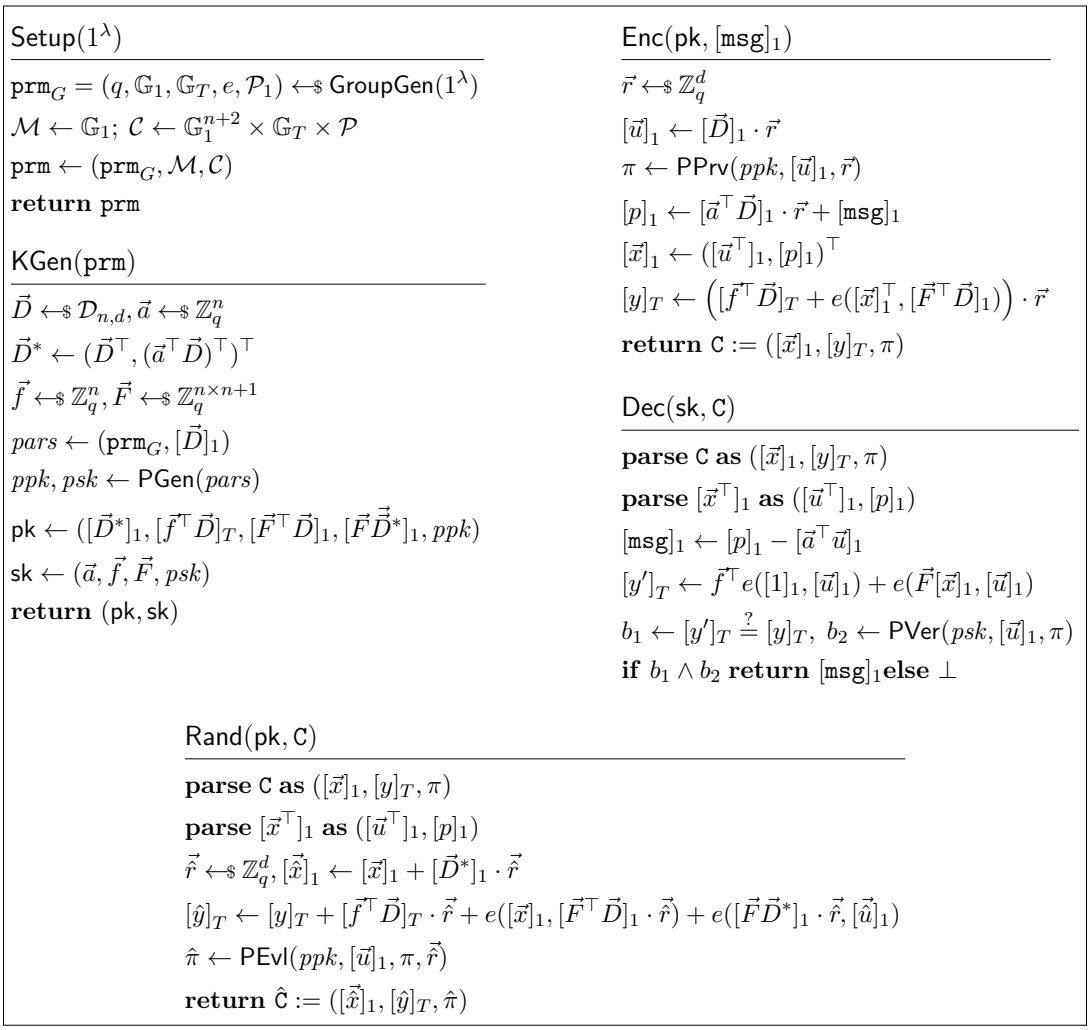


Figure 7.4: Rand-RCCA PKE scheme PKE based on the $\mathcal{D}_{n,d}$-MDDH assumption in type-1 bilinear groups. $\mathcal{P}$ is the support of the proofs for **PS**.

Proof.

$$\begin{aligned}(\vec{f}^\top + \vec{\hat{x}}\vec{F}^\top)\vec{\hat{u}} &= (\vec{f}^\top + (\vec{x} + \vec{D}^*\vec{\hat{r}})^\top \vec{F}^\top)(\vec{u} + \vec{D}\vec{\hat{r}}) \\ &= (\vec{f}^\top + \vec{x}^\top \vec{F}^\top)\vec{u} + (\vec{f}^\top + \vec{x}^\top \vec{F}^\top)(\vec{D}\vec{\hat{r}}) + (\vec{D}^*\vec{\hat{r}})^\top \vec{F}^\top \vec{u} + (\vec{D}^*\vec{\hat{r}})^\top \vec{F}^\top \vec{D}\vec{\hat{r}}\end{aligned}$$

Notice that the third term can be rewritten as:

$$(\vec{D}^*\vec{\hat{r}})^\top \vec{F}^\top \vec{u} = (\vec{F}\vec{D}^*\vec{\hat{r}})^\top \vec{u}$$

□

Indistinguishability. First, we want to focus on property (1) that says that the re-randomization of an honest encryption is identically distributed to a fresh encryption.

Let $\mathtt{C} := ([\vec{x}]_1, [y]_1, \pi)$ be an encryption of $[\mathtt{msg}]_1$ with randomness fixed to $\vec{r}$, and let $\hat{\mathtt{C}} := ([\vec{\hat{x}}]_1, [\hat{y}]_T, \hat{\pi}) \leftarrow\$ \mathsf{Rand}(\mathsf{pk}, \mathtt{C})$ be its re-randomization with randomness $\vec{\hat{r}}$. We show that $\hat{\mathtt{C}}$ is

identically distributed to a fresh encryption of $[\mathtt{msg}]_1$ with randomness $(\vec{r}+\vec{\hat{r}})$. Notice that for any vector $\vec{r}$, the random variable $(\vec{r}+\vec{\hat{r}})$ is uniformly distributed.

It is straightforward to verify that this holds for the element $[\vec{\hat{x}}]_1$, indeed:

$$\vec{\hat{x}} = \vec{D}^*(\vec{r}+\vec{\hat{r}}) + (\vec{0}^\top, \mathtt{msg})^\top.$$

The proof that $[\hat{y}]_T$ is correctly distributed, follows easily from Lemma 7.6.1. What is left to prove is that the element $\hat{\pi}$ is also correctly distributed. We recall that $\hat{\pi}$ is the result of $\mathsf{PEvl}(ppk, [\vec{D}\vec{r}]_1, \pi, \vec{\hat{r}})$. Because of the correctness of the proof system **PS**, it holds that $\hat{\pi}$ is identical to $\mathsf{PPrv}(ppk, [\vec{D}(\vec{r}+\vec{\hat{r}})]_1, (\vec{r}+\vec{\hat{r}}))$ that is exactly the distribution of $[y]_T$ computed by $\mathsf{Enc}(ppk, \mathtt{msg}; (\vec{r}+\vec{\hat{r}}))$.

Correctness. Next, we prove the second property of Definition 2.4.1, namely that for any ciphertext $\mathtt{C}$, it holds that $\mathsf{Dec}(\mathsf{sk}, \mathtt{C}) = \mathsf{Dec}(\mathsf{sk}, \mathtt{C}')$, where $\mathtt{C}'$ is the output of a valid re-randomization of $\mathtt{C}$, with some randomness $\vec{\hat{r}}$.

First, we notice that Rand adds to $\vec{x}$ an encryption of $[0]_1$, therefore if the third line of $\mathsf{Dec}(\mathsf{sk}, \mathtt{C})$ computes $[\mathtt{msg}]_1$, the same happens in $\mathsf{Dec}(\mathsf{sk}, \mathtt{C}')$. Second, if $\mathtt{C}$ is valid, then the correctness follows from the proof given above: $\mathtt{C}'$ is also valid and $\mathsf{Dec}(\mathsf{sk}, \mathtt{C}) = \mathsf{Dec}(\mathsf{sk}, \mathtt{C}')$. Finally, we are left with proving that if $\mathsf{Dec}(\mathsf{sk}, \mathtt{C}) = \bot$, then also $\mathsf{Dec}(\mathsf{sk}, \mathtt{C}')$ returns $\bot$. Assume by contradiction that $\mathsf{Dec}(\mathsf{sk}, \mathtt{C}) = \mathtt{msg} \neq \bot$. Then it must be true that $[\hat{y}]_T = (\vec{f} + \vec{F}\vec{\hat{x}})^\top [\vec{\hat{u}}]_1$ and $\hat{\pi} = \vec{k}^\top([\vec{\hat{u}}]_1 \otimes [\vec{\hat{u}}]_1)$. For the same proof given above, and for the correctness of the benign proof system **PS**, we obtain that $[y]_T = (\vec{f} + \vec{F}\vec{x})^\top [\vec{u}]_1$, $\pi = \vec{k}^\top([\vec{u}]_1 \otimes [\vec{u}]_1)$, and thus $\mathsf{Dec}(\mathsf{sk}, \mathtt{C}) = \mathtt{msg} \neq \bot$, that results in a contradiction.

Tightness of Decryption. The last property we need to prove is the third one of Definition 2.4.1. We observe that, given a vector $[\vec{u}]_1$, there exists a unique value $\pi := \vec{k}^\top([\vec{u}]_1 \otimes [\vec{u}]_1)$ such that $\mathsf{PVer}(psk, [\vec{u}]_1, \pi) = 1$. Assume there exists an adversary who is able to produce a ciphertext that does not decrypt to $\bot$, yet it is not in the encryption range of the scheme, namely there exists no message $\mathtt{msg}$, and no randomness $\vec{r}$ such that $\mathsf{Enc}(\mathsf{pk}, \mathtt{msg}; \vec{r})$ outputs $\mathtt{C}$. Given the above observation, it must be the case that the first component of the ciphertext, i.e. $[\vec{u}]_1$, is not in the span of $[\vec{D}]_1$: in [FFHR19] it is proven that the probability that an adversary succeeds in forging a valid $[y]_T$ in such case, is lower or equal than $\frac{1}{q}$.

Security. We prove that the security of the scheme reduces to the $\mathcal{D}_{n,d}$-MDDH assumption. Below we state the main theorem.

Theorem 7.6.1. *For every PPT adversary $\mathcal{A}$ that makes at most Q_{Enc} encryption and Q_{Dec} decryption queries, there exist adversaries $\mathcal{B}^{\mathrm{mddh}}$, $\mathcal{B}^{\mathrm{snd}}$ with similar running time $T(\mathcal{B}^{\mathrm{mddh}}) \approx T(\mathcal{B}^{\mathrm{snd}}) \approx T(\mathcal{A}) + (Q_{\mathsf{Enc}} + Q_{\mathsf{Dec}}) \cdot \mathsf{poly}(\lambda)$, where $\mathsf{poly}(\lambda)$ is a polynomial independent of $T(\mathcal{A})$, and such that*

$$\begin{aligned}\mathbf{Adv}^{\mathtt{RCCA}}_{\mathcal{A},\mathsf{PKE}}(\lambda) \leq\ & O\left(d \log Q_{\mathsf{Enc}}\right) \cdot \mathbf{Adv}^{\mathtt{MDDH}}_{\mathbb{G}_1,\mathcal{D}_{n,d},\mathcal{B}^{\mathrm{mddh}}}(\lambda) \\ & + \log Q_{\mathsf{Enc}} \cdot \mathbf{Adv}^{\mathtt{snd}}_{\mathcal{B}^{\mathrm{snd}},\mathbf{PS}}(\lambda) + O\left(\tfrac{n^2 Q_{\mathsf{Dec}} Q_{\mathsf{Enc}} \log Q_{\mathsf{Enc}}}{q}\right).\end{aligned}$$

Proof. We give a proof only for the single-user, multi-ciphertext case, i.e., when the adversary

calls the key generation oracle only once. The proof can be easily generalized[4] to the multi-user case almost equivalently to [BBM00, GHK17].

To simplify the notation, since we are in the single-user setting, we omit the index j (which specifies the user) from both encryption and decryption queries. We let $\mathbf{G}_0$ be the $\mathbf{Exp}^{\mathtt{mRCCA}}_{\mathcal{A},\mathsf{PKE}}$ experiment, and we denote with ϵ_i the advantage of $\mathcal{A}$ to win $\mathbf{G}_i$, i.e. $\epsilon_i := |\Pr[\mathbf{G}_i = 1] - \frac{1}{2}|$.

The games keep track of the number of challenge ciphertexts produced. Specifically, let ctr be a variable that counts the number of challenge ciphertexts output by the encryption oracle: ctr is set to 0 at the beginning of the games and, whenever the adversary calls the encryption oracle, it is increased.

Game $\mathbf{G}_1$. This game is identical to the previous one, but the encryption oracle computes the values $[y]_T$ and $[p]$ using secret keys (instead of public keys). Specifically, upon the j-th query to the encryption oracle, the game computes the ciphertext $\mathtt{C}_j = ([\vec{x}_j], [y_j]_T, \pi_j)$ as described by the encryption procedure, but where we compute $[y_j]_T \leftarrow \vec{f}^\top[\vec{u}_j] + [\vec{x}_j]^\top \vec{F}^\top[\vec{u}_j]$ and $[p_j] \leftarrow \vec{a}^\top[\vec{u}_j] + [\mathtt{msg}_{j,b^*}]$. By linearity, this game is perfectly equivalent to the previous one, thus $\epsilon_1 = \epsilon_0$.

Game $\mathbf{G}_2$. This game is identical to the previous one, but the encryption oracle simulates the benign proofs π. We rely on the perfect zero-knowledge of the benign proof system. The reduction is standard, therefore we omit it. Since the proof system satisfies perfect zero-knowledge we have $\epsilon_2 = \epsilon_1$.

Game $\mathbf{G}_3$. At the very beginning, the game additionally samples matrices $\vec{\vec{D}}_b \leftarrow\!\!\$\ \mathbb{Z}_q^{n\times d}$ for $b \in \{0,1\}$, and sets $\vec{D}_b \leftarrow \left(\vec{D} | \vec{\vec{D}}_b\right)$. The encryption oracle in this game samples $[\vec{u}]$ from the span of $[\vec{D}_0]$. We apply a standard reduction to the Q_{Enc}-fold $\mathcal{D}_{n,d}$-MDDH assumption, twice, and we prove that no adversary can distinguish this game from the previous one: we first tightly switch the vectors in the challenge ciphertexts from the span of $[\vec{D}]$ to uniformly random vectors of $\mathbb{G}_1^n$; next, we use the Q_{Enc}-fold $\mathcal{D}_{n,2d}$-MDDH assumption to switch these vectors from random to the span of $[\vec{D}_0]$. To be formal, we build adversaries $\mathcal{B}$, $\mathcal{B}'$ such that for a polynomial $p(\lambda)$ independent of $T(\mathcal{A})$, we have $T(\mathcal{B}) \approx T(\mathcal{B}') \approx T(\mathcal{A}) + (Q_{\mathsf{Enc}} + Q_{\mathsf{Dec}}) \cdot p(\lambda)$ and

$$|\epsilon_3 - \epsilon_2| \leq \mathbf{Adv}^{\mathtt{Q_{Enc}-MDDH}}_{\mathbb{G}_1,\mathcal{D}_{n,d},\mathcal{B}}(\lambda) + \mathbf{Adv}^{\mathtt{Q_{Enc}-MDDH}}_{\mathbb{G}_1,\mathcal{D}_{n,2d},\mathcal{B}'}(\lambda)$$

Let $([\vec{D}], [\vec{V}])$ be the Q_{Enc}-fold $\mathcal{D}_{n,d}$-MDDH challenge received by $\mathcal{B}$, with $[\vec{D}] \in \mathbb{G}_1^{n\times d}$ and $[\vec{V}] := ([\vec{v}_1], \ldots, [\vec{v}_{Q_{\mathsf{Enc}}}]) \in \mathbb{G}_1^{n\times Q_{\mathsf{Enc}}}$. Then, $\mathcal{B}$ samples the secret material $\vec{a} \leftarrow\!\!\$\ \mathbb{Z}_q^n$, $\vec{f} \leftarrow\!\!\$\ \mathbb{Z}_q^n, \vec{F} \leftarrow\!\!\$\ \mathbb{Z}_q^{n\times n+1}$, generates the keys for the benign proof system, running $\mathsf{PGen}([\vec{D}])$, and finally sends the public key to $\mathcal{A}$. On the j-th query to the encryption oracle, $\mathcal{B}$ sets

$$[\vec{u}_j] \leftarrow [\vec{v}_j]$$
$$[y_j]_T \leftarrow \vec{f}^\top[\vec{u}_j]_T + [\vec{x}_j]^\top \vec{F}^\top[\vec{u}_j]$$

$\mathcal{B}$ has generated the secret key itself, so it can perfectly simulate the decryption oracle. In case $[\vec{V}] = [\vec{DR}]$, $\mathcal{B}$ perfectly simulates $\mathbf{G}_2$. In case $[\vec{V}]$ is uniformly random over $\mathbb{G}_1^{n\times Q_{\mathsf{Enc}}}$, $\mathcal{B}$

[4] We rely on the self-reducibility of the MDDH assumption: in particular, we can generate m different matrices $\vec{D}_j$ (one for each user) from one single challenge of the (many-fold) MDDH assumption and adapt accordingly the ciphertexts, namely, by mapping the ciphertext for the j-th user through the same linear transformation that maps the MDDH-challenge matrix to the matrix $\vec{D}_j$.

simulates an intermediary game $\mathbf{H}$. Analogously, we can build an adversary $\mathcal{B}'$ on the Q_{Enc}-fold $\mathcal{D}_{n,2d}$-MDDH assumption, who simulates $\mathbf{H}$ if $[\vec{V}]$ is uniformly random over $\mathbb{G}_1^{n\times Q_{\mathsf{Enc}}}$, and game $\mathbf{G}_3$ if $[\vec{V}] = [\vec{D}_0\vec{R}]$. Altogether, this proves the claim stated above: from Lemma 2.3.2 and Corollary 2.3.1, we obtain an adversary $\mathcal{B}''$ such that $T(\mathcal{B}'') \approx T(\mathcal{A}) + (Q_{\mathsf{Enc}} + Q_{\mathsf{Dec}}) \cdot \mathsf{poly}(\lambda)$ where $\mathsf{poly}(\lambda)$ is independent of $T(A)$ and

$$|\epsilon_3 - \epsilon_2| \leq 2(n-d)\mathbf{Adv}^{\mathtt{mddh}}_{\mathbb{G}_1,\mathcal{D}_{n,d},\mathcal{B}''}(\lambda) + \frac{2}{q-1}.$$

Game $\mathbf{G}_4$. In this experiment, we add an explicit check to the decryption oracle. First recall that $\vec{D}^*$ is defined in Fig. 7.4 as the matrix whose first n rows are equal to the matrix $\vec{D}$ and last row is equal to $\vec{a}^\top\vec{D}$. Upon query $\mathtt{C} := ([\vec{x}], [y]_T, \pi)$ to the decryption oracle, where $[\vec{x}]^\top := ([\vec{u}]^\top, [p])$, the oracle additionally checks that:

$$\vec{u} \in span(\vec{D}) \vee \exists j : \vec{D^*}^\perp \vec{x}_j = \vec{D^*}^\perp \vec{x} \tag{7.2}$$

where $\vec{D^*}^\perp \vec{D^*} = 0$, and $\mathcal{Q}_{\mathsf{Enc}} = \{\mathtt{C}_j = ([\vec{x}_j], [y_j]_T, \pi_j) : j \leq [\mathsf{ctr}]\}$ is the set of challenge ciphertexts. If the condition holds, the decryption oracle proceeds by running the decryption procedure as usual, otherwise it returns $\perp$ to the adversary. We notice that the new condition can be checked efficiently since we know $\vec{D} \in \mathbb{Z}_q^{n\times d}$ and $\vec{a} \in \mathbb{Z}_q^n$.

The distinguishing event between $\mathbf{G}_4$ and $\mathbf{G}_3$ is that the adversary queries the decryption oracle with a ciphertext that would not decrypt to $\perp$ (according to the original decryption rules of $\mathbf{G}_3$), but where Eq. (7.2) holds. We call such query to the decryption oracle a "critical query", i.e. a decryption query where:

- $[\vec{u}] \notin span([\vec{D}])$ and $\forall j : \vec{D^*}^\perp \vec{x}_j \neq \vec{D^*}^\perp \vec{x}$ (the latter condition implies that $[\vec{u}]$ is not the result of an honest re-randomization of a previous challenge ciphertext)
- the proof π is valid, and $[y]_T = \vec{f}^\top[\vec{u}]_T + [\vec{x}]^\top\vec{F}^\top[\vec{u}]$, i.e., the consistency check holds.

In other words, a critical decryption query consists of a ciphertext that does not decrypt to $\perp$ (according to the original decryption rules), is not a re-randomization of a challenge ciphertext, but has a $[\vec{u}]$ that could not have been generated in an "honest" encryption. For this step, we refer to Lemma 7.6.2.

Game $\mathbf{G}_5$. This game is equivalent to $\mathbf{G}_4$, but we modify the rules of the decryption oracle once again. For any j, let $\mathtt{msg}_{j,0}$ and $\mathtt{msg}_{j,1}$ be the challenge messages queried by $\mathcal{A}$ at the j-th query to the encryption oracle. Upon decryption oracle query $\mathtt{C} = ([\vec{x}], [y]_T, \pi)$, if $\exists j : \vec{D^*}^\perp \vec{x}_j = \vec{D^*}^\perp \vec{x}$ where recall $\mathcal{Q}_{\mathsf{Enc}} = \{([\vec{x}_j], [y_j], \pi_j) : j \leq \mathsf{ctr}\}$, and both the proof π verifies and the consistency check holds, then the decryption oracle immediately returns the symbol $\diamond_{\mathcal{J}}$ where $\mathcal{J} \leftarrow \mathcal{Q}.\mathrm{find}(\mathtt{msg}_{j,0})$.

Notice that we can rewrite the decryption procedure as $\mathtt{msg} = (-\vec{a}^\top, 1)[\vec{x}]$. We observe that the vector $(-\vec{a}^\top, 1)$ is in the span of $\vec{D^*}^\perp$, since it holds that $(-\vec{a}^\top, 1)\vec{D^*} = -\vec{a}^\top\vec{D} + \vec{a}^\top\vec{D} = 0$. Thus, at any decryption query, if $\vec{D^*}^\perp \vec{x}_j = \vec{D^*}^\perp \vec{x}_j$ for some challenge ciphertext $\mathtt{C}_j$ then $(-\vec{a}^\top, 1)[\vec{x}_j] = (-\vec{a}^\top, 1)[\vec{x}]$, and therefore the decryption oracle would compute the message $\mathtt{msg}_{j,b^*}$ and output the symbol $\diamond_{\mathcal{J}}$, where $\mathcal{J} = \mathcal{Q}.\mathrm{find}(\mathtt{msg}_{j,b^*})$. Moreover, notice that

$\mathcal{Q}.\text{find}(\mathtt{msg}_{j,b^*}) = \mathcal{Q}.\text{find}(\mathtt{msg}_{j,0})$ by definition of the security experiment. This shows that $\epsilon_5 = \epsilon_4$.

Game $\mathbf{G}_6$. In this last step, we encrypt random messages. Formally, at the j-th encryption query the oracle (on input messages $\mathtt{msg}_{j,0}, \mathtt{msg}_{j,1}$) encrypts the message $\mathtt{msg}_{j,b^*} + \mathsf{R}_j$, where R_j is random. Clearly, it holds that $\epsilon_6 = 0$ as in fact, because of the change introduced in $\mathbf{G}_6$, the ciphertexts are independent of the challenge bit b^*, and, by the changes introduced in $\mathbf{G}_4$ and $\mathbf{G}_5$, the decryption queries are independent of the challenge bit. We prove that $\mathbf{G}_5$ and $\mathbf{G}_6$ are indistinguishable.

We first observe switching from $\vec{a}^\top$ to $\vec{a}^\top + \vec{r}^\top \vec{D}^\perp$ for a random $\vec{r}$ and using the vector $\vec{a}^\top + \vec{r}^\top \vec{D}^\perp$ both for the encryption and decryption oracle queries does not change the view of the adversary. So consider a new game $\mathbf{G}'_5$, identical to $\mathbf{G}_5$ but where we set $\vec{a}^\top + \vec{r}^\top \vec{D}^\perp$ instead of $\vec{a}$.

Let $[\vec{u}_j] = [\vec{D}_0]\vec{r}_j$ be the first component of the j-th challenge ciphertext, computed by the encryption oracle. It holds that $[p_j] = (\vec{a}^\top + \vec{r}^\top \vec{D}^\perp)[\vec{u}_j] + [\mathtt{msg}_{jb^*}]$, which is equal to $\vec{a}^\top[\vec{u}_j] + [\mathtt{msg}_{jb^*}] + \vec{r}^\top(\vec{D}^\perp[\vec{D}_0])\vec{r}_j$. We show that the term $\vec{r}^\top(\vec{D}^\perp[\vec{D}_0])$, is statistically close to a random vector $[\vec{v}]$. The reason is that for any fixed choice of the columns of $\vec{D}_0$, the rows of $\vec{D}^\perp$ are linearly independent with overwhelming probability (over $\vec{D}^\perp$).

We now reduce to MDDH to show that $|\epsilon_6 - \epsilon_{5'}|$ is negligible in the security parameter for any adversary $\mathcal{A}$: formally, we build an adversary $\mathcal{B}$ such that $T(\mathcal{B}) \approx T(\mathcal{A}) + (Q_{\mathsf{Enc}} + Q_{\mathsf{Dec}}) \cdot \mathsf{poly}(\lambda)$ and:

$$|\epsilon_6 - \epsilon_{5'}| \leq \mathbf{Adv}^{\mathtt{mddh}}_{\mathbb{G}_1, \mathcal{U}_{2d+1,2d}, \mathcal{B}}(\lambda).$$

Let $[\vec{B}] \in \mathbb{G}_1^{2d+1 \times 2d}$, and let $([\vec{h}_1], \ldots, [\vec{h}_{Q_{\mathsf{Enc}}}]) \in \mathbb{G}_1^{2d+1 \times Q_{\mathsf{Enc}}}$ be the $\mathcal{U}_{2d+1,2d}$-MDDH challenge received in input. We indicate the upper values of $[\vec{h}_j]$ as $\overline{[\vec{h}_j]} \in \mathbb{G}_1^{2d}$, and we indicate its lower values as $\underline{[\vec{h}_j]} \in \mathbb{G}_1$. In the reduction, the encryption oracle sets $[\vec{u}_j] \leftarrow \vec{D}_0 \overline{[\vec{h}_j]}$, which implicitly sets $\vec{r}_j = \overline{\vec{h}_j}$. Also, it sets $[p_j] = [\mathtt{msg}_{jb^*}] + \vec{a}^\top[\vec{u}_j] + \underline{[\vec{h}_j]}$. If $([\vec{B}], [\vec{h}_1], \ldots, [\vec{h}_Q])$ is a real MDDH challenge, $\mathcal{B}$ simulates the game $\mathbf{G}_{5'}$. Indeed, it must hold that $\vec{h}_j = \vec{B}\vec{s}_j$, for some $\vec{s}_j \in \mathbb{Z}_q^{2d}$. Also, let $\overline{[\vec{B}]}$ be the upper square matrix of $[\vec{B}]$, and let $\underline{[\vec{B}]}$ be the last row of $[\vec{B}]$. $\underline{[\vec{h}_j]} = \underline{[\vec{B}]}\vec{s}_j = \underline{[\vec{B}]}\overline{\vec{B}}^{-1}\vec{r}_j$. And we have that the distribution of $\underline{[\vec{B}]}\overline{\vec{B}}^{-1}$ is statistically close to a random element $\vec{v}$. Otherwise, $\mathcal{B}$ simulates the game $\mathbf{G}_6$. Finally, by applying Lemma 2.3.2, Lemma 2.3.3 and Lemma 2.3.4, we can always build a new adversary $\mathcal{C}$ such that $T(\mathcal{C}) \approx T(\mathcal{A}) + (Q_{\mathsf{Enc}} + Q_{\mathsf{Dec}}) \cdot \mathsf{poly}(\lambda)$ and:

$$|\epsilon_6 - \epsilon_5| = |\epsilon_6 - \epsilon_{5'}| \leq (n-d)\mathbf{Adv}^{\mathtt{mddh}}_{\mathbb{G}_1, \mathcal{D}_{n,d}, \mathcal{C}}(\lambda) + \frac{1}{q-1}.$$

□

Lemma 7.6.2. *The security games $\mathbf{G}_3$ and $\mathbf{G}_4$ defined for the proof of Theorem 7.6.1 (security of the RCCA symmetric scheme, see Fig. 7.4) are computationally indistinguishable. For any PPT adversary $\mathcal{A}$, we build PPT adversaries $\mathcal{B}$, $\mathcal{B}'$ with running times similar to $\mathcal{A}$ such that:*

$$\begin{aligned}|\epsilon_3 - \epsilon_4| \leq\ & O(d \log Q_{\mathsf{Enc}})\mathbf{Adv}^{\mathtt{MDDH}}_{\mathbb{G}_1, \mathcal{D}_{n,d}, \mathcal{B}}(\lambda) + \log Q_{\mathsf{Enc}}\mathbf{Adv}^{\mathtt{snd}}_{\mathcal{B}'}(\lambda) \\ & + O\left(\tfrac{n^2 Q_{\mathsf{Dec}} Q_{\mathsf{Enc}} \log Q_{\mathsf{Enc}}}{q}\right)\end{aligned}$$

Proof. We denote the probability that the adversary $\mathcal{A}$ wins game $\mathbf{H}_x$ by $\epsilon_{\mathbf{H}_x}$. In the following, we will bound $\epsilon_{\mathbf{H}_0}$ via a sequence of games.

Hybrid $\mathbf{H}_0$. This hybrid is the same as $\mathbf{G}_3$ but immediately outputs 1 if the adversary makes a "critical query". Specifically, the hybrid executes $\mathbf{G}_3$ but the decryption oracle upon input $\mathtt{C}$ parses it as $([\vec{x}], [y]_T, \pi)$ and checks that Eq. (7.2) holds; if it holds, the decryption oracle continues as before. Otherwise, returns the message "critical", and $\mathbf{H}_0$ stops the interaction, immediately returning 1. Since the hybrid outputs 1 when the distinguishing event between $\mathbf{G}_3$ and $\mathbf{G}_4$ happens, we have that $|\epsilon_3 - \epsilon_4| \leq \epsilon_{\mathbf{H}_0}$. Also notice that the checks in Eq. (7.2) can be efficiently performed given the knowledge of the matrix $\vec{D}$.

Hybrid $\mathbf{H}_1$. This hybrid is preparatory for the next one. We inject randomness into the encryption/decryption keys, adding a vector $(\vec{z}\vec{D}^{\perp})$ to the secret key $\vec{f}^{\top}$, common to all the encryption queries, where $\vec{z} \in \mathbb{Z}_q^{n-d}$. Specifically, at the very beginning of the experiment we sample the vector $\vec{z} \leftarrow\!\!\$\ \mathbb{Z}_q^{n-d}$, we sample $\vec{f}$ and compute the public key material $[\vec{f}^{\top}\vec{D}]$ and moreover:

- The encryption oracle, at the j-th query, computes the values $[y_j]_T$ as follows:

$$[y_j]_T \leftarrow (\vec{f}^{\top} + \vec{z}\vec{D}^{\perp})[\vec{u}_j]_T + [\vec{x}_j]^{\top}\vec{F}^{\top}[\vec{u}_j]$$

- Similarly, the decryption oracle, upon input the ciphertext $\mathtt{C} = ([\vec{x}], [y]_T, \pi)$ computes the bit b_1 (i.e. the bit of the consistency check) by computing the value $[y']_T$ and checking if $[y]_T \stackrel{?}{=} [y']_T$ where $[y']_T$ is computed as:

$$[y']_T \leftarrow (\vec{f}^{\top} + \vec{z}\vec{D}^{\perp})[\vec{u}]_T + [\vec{x}]^{\top}\vec{F}^{\top}[\vec{u}]$$

These new rules do not change the view of the adversary since both $\vec{f}^{\top}$ and $\vec{f}^{\top} + \vec{z}\vec{D}^{\perp}$ are uniformly distributed over $\mathbb{Z}_q^{1\times n}$ given the public key material $[\vec{f}^{\top}\vec{D}]$. Thus, we obtain $\epsilon_{\mathbf{H}_1} = \epsilon_{\mathbf{H}_0}$.

Hybrid $\mathbf{H}_2$. Let $P : \{0,1\}^* \to \mathbb{Z}_q^{1\times n-d}$ be a uniformly random function. In this hybrid we use the following rules for encryption and decryption:

- The encryption oracle, at the j-th query, computes the values $[y_j]_T$ as follows:

$$[y_j]_T \leftarrow (\vec{f}^{\top} + P(j)\,\vec{D}^{\perp})[\vec{u}_j]_T + [\vec{x}_j]^{\top}\vec{F}^{\top}[\vec{u}_j]$$

- For each decryption oracle query, we first define a set $\mathcal{S}$ over which the decryption oracle iterates to test the consistency check. The definition of the set $\mathcal{S}$ is carefully crafted to define the behavior of the hybrid experiment in case of *replay attack* from the adversary [5].

[5]The reader might have notice that this is where our proof strategy needs to differentiate from the original partitioning technique from [GHK17]. In particular, we conclude the proof of Lemma 7.6.2 (see pag. 215) by reducing to Lemma 7.3.1, the reduction will know all the secret keys but one for a (uniformly sampled) random index.

Recall that ctr counts the number of challenge ciphertexts output by the encryption oracle and that $\mathcal{Q}_{\mathsf{Enc}} = \{\mathtt{C}_j = ([\vec{x}_j], [y_j]_T, \pi_j) : j \leq \mathsf{ctr}\}$. Upon input the ciphertext $\mathtt{C} = ([\vec{x}], [y]_T, \pi)$, the decryption oracle first sets:

$$\begin{aligned} &\mathcal{S} := \{j\} && \text{if } \exists j \leq \mathsf{ctr} : \vec{D}^{*\perp}[\vec{x}] = \vec{D}^{*\perp}[\vec{x}_j] \\ &\mathcal{S} := \{j : j \leq \mathsf{ctr}\} && \text{otherwise} \end{aligned}$$

then it computes the bit b_1 (i.e. the bit of the consistency check for $\mathtt{C}$, see Fig. 7.4) differently by checking that

$$\exists j \in \mathcal{S} : [y]_T \stackrel{?}{=} (\vec{f}^\top + P(j)\,\vec{D}^\perp)[\vec{u}]_T + [\vec{x}]^\top \vec{F}^\top [\vec{u}].$$

Moving from $\mathbf{H}_1$ to $\mathbf{H}_2$ requires a series of hybrids $\mathbf{H}_{1,i,i'}$, $i \in [\log(Q_{\mathsf{Enc}})], i' \in [6]$.

Hybrid $\mathbf{H}_{1,i,0}$. Let P_i be a random function that takes in input strings of length i (for $i = 0$, we can imagine this as a constant function defined on the empty string) and returns row vectors of length $n - d$.

- On input the j-th query, the encryption oracle samples $[\vec{u}_j]$ from the span of $[\vec{D}_0]$. The element $[y_j]_T$ is computed as

$$[y_j]_T \leftarrow (\vec{f} + P_i(j_{|i})\,\vec{D}^\perp)[\vec{u}_j] + [\vec{x}_j]^\top \vec{F}^\top [\vec{u}_j].$$

- Upon input the ciphertext $\mathtt{C} = ([\vec{x}], [y]_T, \pi)$, define:

$$\begin{aligned} &\mathcal{S} := \{j_{|i}\} && \text{if } \exists j \leq \mathsf{ctr} : \vec{D}^{*\perp}[\vec{x}] = \vec{D}^{*\perp}[\vec{x}_j] \\ &\mathcal{S} := \{j_{|i} : j \leq \mathsf{ctr}\} && \text{otherwise} \end{aligned}$$

  it then executes the same code of the previous hybrid.

When $i = 0$, for any value j the string $j_{|0}$ is equal to the empty string, thus, in $\mathbf{H}_{1,0,0}$, the random function P_0 is always called on input the empty string. In particular, either when $\vec{D}^{*\perp}[\vec{x}] = \vec{D}^{*\perp}[\vec{x}_j]$ holds or when it does not, the consistency check performed is exactly the same. Thus, the difference between hybrid $\mathbf{H}_{1,0,0}$ and $\mathbf{H}_1$ is only syntactical.

Hybrid $\mathbf{H}_{1,i,1}$. This hybrid is equivalent to the previous one, but here the encryption oracle, on input the j-th query, generates $[\vec{u}_j]$ in the span of $[\vec{D}_{j[i+1]}]$. We rely on the MDDH assumption to prove indistinguishability between the two hybrids. We proceed in two steps:

- We first switch the j-th vector $[\vec{u}_j]$ computed by the encryption oracle to a vector in the span of $[(\vec{D}|\vec{U})]$, where $\vec{U}$ is uniform over $\mathbb{Z}_q^{n \times d}$, if the $(i+1)$-th bit of the binary representation of j is equal to 1. We call this intermediate hybrid $\mathbf{H}_{A_i}$.

- Finally, we switch the j-th vector $[\vec{u}_j]$ computed by the encryption oracle to a vector in the span of $[(\vec{D}|\vec{D}_1)] = [\vec{D}_1]$, if the $(i+1)$-th bit of the binary representation of j is equal to 1.

First we show indistinguishability between $\mathbf{H}_{1,i,0}$ and $\mathbf{H}_{A_i}$. Let $\mathcal{B}_A$ be an MDDH adversary receiving the Q_{Enc}-fold $\mathcal{D}_{n,d}$-MDDH challenge $([\vec{\vec{D}}_0], [\vec{h}_1], \ldots, [\vec{h}_{Q_{\mathsf{Enc}}}])$ as input. $\mathcal{B}_A$ can sample a random matrix $\vec{D} \leftarrow\!\!\$\ \mathcal{D}_{n,d}$, a random matrix $\vec{\vec{D}}_1 \in \mathbb{Z}_q^{n\times d}$, the secret material $\vec{a} \leftarrow\!\!\$\ \mathbb{Z}_q^n$, $\vec{f} \leftarrow\!\!\$\ \mathbb{Z}_q^d, \vec{F} \leftarrow\!\!\$\ \mathbb{Z}_q^{n\times n+1}$ and the secret material for the benign proof system (since $\mathcal{B}_A$ knows $\vec{D}$, this can be easily achieved running $\mathsf{PGen}([\vec{D}])$). Finally, $\mathcal{B}_A$ samples a challenge bit b and gives the public key of the scheme to $\mathcal{A}$. $\mathcal{B}_A$ simulates the encryption oracle as follows. On input the j-th pair of messages $(\mathtt{msg}_0, \mathtt{msg}_1)$:

- if the $(i+1)$-th bit of the binary representation of j is equal to 0, the adversary sets $[\vec{u}_j] \leftarrow [\vec{D}_0]\vec{r}_j$,
- else, samples a random vector $\vec{r} \in \mathbb{Z}_q^d$, and computes $[\vec{u}_j] \leftarrow [\vec{D}]\vec{r} + [\vec{h}_j]$.

Note that $\mathcal{B}_A$ can still simulate the decryption oracle, because of the knowledge of the secret material $\vec{a}, \vec{f}, \vec{F}$ and of the matrix $\vec{D}$. Since $\mathcal{B}_A$ knows both the matrix $\vec{D}$ and the vector $\vec{a}$, can always find a matrix $\vec{D}^{*\perp}$ such that $\vec{D}^{*\perp}\vec{D}^* = 0$. This allows $\mathcal{B}_A$ to catch critical queries. If the tuple is a real MDDH tuple, i.e. $[\vec{h}_j] = [\vec{\vec{D}}_0]\vec{r}_j$, the game described is perfectly equivalent to $\mathbf{H}_{1,i,0}$. Otherwise, if the challenge vectors are uniformly random, the game simulated is equivalent to $\mathbf{H}_{A,i}$. The next step is to switch the j-th vector $[\vec{u}_j]$ computed by the encryption oracle to a vector in the span of $[(\vec{D}|\vec{\vec{D}}_1)] = [\vec{D}_1]$ if the $(i+1)$-th bit of the binary representation of j is equal to 1. This transformation is similar to the previous one, therefore we omit the details. Altogether, combining the previous adversaries, and considering Lemma 2.3.2 and Corollary 2.3.1, we obtain an adversary $\mathcal{C}$ such that:

$$|\epsilon_{\mathbf{H}_{1,i,1}} - \epsilon_{\mathbf{H}_{1,i,0}}| \leq 2(n-d)\mathbf{Adv}^{\mathtt{mddh}}_{\mathbb{G}_1,\mathcal{D}_{n,d},\mathcal{C}}(\lambda) + \frac{2}{q-1}.$$

Hybrid $\mathbf{H}_{1,i,2}$. We add an explicit check to the decryption oracle. Specifically, at each decryption oracle query the hybrid additionally checks if $\vec{u} \notin span(\vec{D}_0) \cup span(\vec{D}_1)$, and if it is the case the decryption oracle returns immediately $\perp$ to the adversary. We rely on the soundness of the underlying benign proof system and the reduction is standard. In particular, the only condition that would allow distinguishing between this hybrid and the previous one is to query the decryption oracle with a ciphertext $\mathtt{C} = ([\vec{x}], [y]_T, \pi)$ where:

- $\vec{u} \notin span(\vec{D}_0) \cup span(\vec{D}_1)$
- the decryption oracle in the hybrid $\mathbf{H}_{1,i,1}$ would not return $\perp$.

For such query it holds that $\mathsf{PVer}(psk, [\vec{u}], \pi) = 1$. We build an adversary $\mathcal{B}$ against the $(\mathcal{L}^{\mathrm{sim}}, \mathcal{L}^{\mathrm{ver}}, \mathcal{L}^{\mathrm{snd}})$-soundness of the proof system. (Recall that $\mathcal{L}^{\mathrm{snd}} = \mathcal{L}^{\mathrm{sim}} = span(\vec{D}_0) \cup span(\vec{D}_1)$, and $\mathcal{L}^{\mathrm{ver}} = \mathbb{Z}_q^n$.)

The adversary $\mathcal{B}$ samples the secret material $\vec{a}, \vec{f}, \vec{F}$; then, it queries the challenger to obtain the public key of the benign proof system, associated with the matrix $\vec{D}$, and finally gives $\mathcal{A}$ all the public key material. The adversary $\mathcal{B}$ can easily simulate the encryption oracle since it knows all the necessary information. To compute the proof π_j associated with the j-th encryption oracle query, it queries the simulation oracle offered by the challenger: it holds

that $\vec{u}_j \in \mathcal{L}^{\text{sim}}$, for all $j \in [Q_{\mathsf{Enc}}]$. When the adversary makes a decryption query, $\mathcal{B}$ needs to verify that the proof π is accepted by PVer; so, it forwards $(\vec{u}, \pi)$ to the challenger. Since $\mathcal{L}^{\text{ver}}$ is equal to $\mathbb{Z}_q^n$, the verification oracle always returns a verdict bit, and $\mathcal{B}$ can proceed in the natural way the simulation of the decryption oracle. At some point $\mathcal{B}$ queries the verification oracle with some $([\vec{u}], \pi)$ such that $\vec{u} \notin span(\vec{D}_0) \cup span(\vec{D}_1)$, i.e., $\vec{u} \notin \mathcal{L}^{\text{snd}}$, but $\mathsf{PVer}(psk, [\vec{u}], \pi) = 1$. This is the event that lets $\mathcal{B}$ win the soundness game. The adversary $\mathcal{B}$ runs in time $T(\mathcal{B}) \approx T(A) + (Q_{\mathsf{Enc}} + Q_{\mathsf{Dec}}) \cdot \mathsf{poly}(\lambda)$, where $\mathsf{poly}(\lambda)$ is a polynomial independent of $T(A)$. Moreover, notice that when the distinguishing event happens the adversary $\mathcal{B}$ wins the soundness game, thus:

$$|\epsilon_{\mathbf{H}_{1,i,2}} - \epsilon_{\mathbf{H}_{1,i,1}}| \leq \mathbf{Adv}^{\mathbf{snd}}_{\mathcal{B},\mathbf{PS}}(\lambda).$$

Hybrid $\mathbf{H}_{1,i,3}$. In this hybrid we increase the entropy of the secret keys during encryption queries.

- The encryption oracle, at the j-th query, computes the values $[y_j]_T$ as follows:

$$[y_j]_T \leftarrow (\vec{f}^{\top} + P_{i+1}(j_{|i+1})\,\vec{D}^{\perp})[\vec{u}_j] + [\vec{x}_j]^{\top}\vec{F}^{\top}[\vec{u}_j].$$

- The decryption oracle, upon input the ciphertext $\mathtt{C} = ([\vec{x}], [y]_T, \pi)$ additionally checks that $\exists d$ s.t. $\vec{u} \in span(\vec{D}_d)$ and in such case it sets:

$$\mathcal{S} := \{j_{|i} \,\|d\} \qquad \text{if } \exists j \leq \mathsf{ctr} : \vec{D}^{*\perp}[\vec{x}] = \vec{D}^{*\perp}[\vec{x}_j]$$
$$\mathcal{S} := \{j_{|i} \,\|d : j \leq \mathsf{ctr}\} \qquad \text{otherwise}$$

  and it continues executing the same code of the previous hybrid.

We prove that $|\epsilon_{\mathbf{H}_{1,i,2}} - \epsilon_{\mathbf{H}_{1,i,3}}|$ is negligible. We first transit to an intermediate hybrid $\mathbf{H}'_i$ where instead of using the function $P_i(\cdot)\vec{D}^{\perp}$, we use the function $P'_i(\cdot) := P_i^{(0)}(\cdot)\vec{D}_0^{\perp} + P_i^{(1)}(\cdot)\vec{D}_1^{\perp}$, where $P_i^{(0)}$ and $P_i^{(1)}$ are two uniformly random functions with domain $\{0,1\}^i$. Notice that $P'_i(\cdot)$ is a uniformly random function that maps strings in $\{0,1\}^i$ to vectors in $rowspan(\vec{D}_0^{\perp}) + rowspan(\vec{D}_1^{\perp})$ while $P_i(\cdot)\vec{D}^{\perp}$ is a uniformly random function that maps string in $\{0,1\}^i$ to vectors in $rowspan(\vec{D}^{\perp})$. Thus, the distinguishing event between $\mathbf{H}_{i,j,2}$ and this intermediate hybrid is the event that $rowspan(\vec{D}_0^{\perp}) + rowspan(\vec{D}_1^{\perp}) \neq rowspan(\vec{D}^{\perp})$. The latter event happens with probability at most $1/q$, in fact the event happens if and only if the subspace $span(\vec{D}_0|\vec{D}_1)$ has dimension strictly less than $2d$ and recall that the columns of such matrices are sampled uniformly at random. Next, we define the function $P_{i+1}^{(b)} : \{0,1\}^{i+1} \to \mathbb{Z}_q^{1\times(n-2d)}$, $\forall b \in \{0,1\}$:

$$P_{i+1}^{(b)}(\mathtt{x}) = \begin{cases} P_i^{(b)}(\mathtt{x}_{|i}), & \mathtt{x}[i+1] \neq b \\ \tilde{P}_i^{(b)}(\mathtt{x}_{|i}), & \text{else} \end{cases}$$

where $P_i, \tilde{P}_i$ are two uniformly (and independent) random functions. Notice that $P_{i+1}^{(b)}$ is a uniformly random function.

We define a second intermediate hybrid $\mathbf{H}'_{i+1}$ where for the encryption oracle queries instead of using the random function P'_i applied to the indexes $j_{|i}$ we use the function P'_{i+1} applied to the indexes $j_{|i+1}$, and for the decryption oracle queries we use P'_{i+1} applied to $(j_{|i}\|d)$, where

d is such that $\vec{u}_j \in span(\vec{D}_d)$ (as described in the $\mathbf{H}_{1,i,3}$). We show that $\mathbf{H}'_i$ and $\mathbf{H}'_{i+1}$ are equivalently distributed. Indeed, in this second intermediate hybrid, at the j-th encryption oracle query we compute

$$[y_j]_T \leftarrow (\vec{f}^\top + P'_{i+1}(j_{|i+1}))[\vec{u}_j] + [\vec{x}_j]^\top \vec{F}^\top [\vec{u}_j].$$

Moreover, we have that $P'_{i+1}(j_{|i+1})\vec{u}_j = P'_i(j_{|i})\vec{u}_j$, in fact:

$$\begin{aligned} P'_{i+1}(j_{|i+1})\vec{u}_j &= \left(P_i^{(1-j[i+1])}(j_{|i})\vec{D}^\perp_{1-j[i+1]} + \tilde{P}_i^{(j[i+1])}(j_{|i})\vec{D}^\perp_{j[i+1]}\right)\vec{D}_{j[i+1]}\vec{r}_j \\ &= \left(P_i^{(1-j[i+1])}(j_{|i})\vec{D}^\perp_{1-j[i+1]}\right)\vec{D}_{j[i+1]}\vec{r}_j \\ &= \left(P_i^{(0)}(j_{|i})\vec{D}^\perp_0 + P_i^{(1)}(j_{|i})\vec{D}^\perp_1\right)\vec{D}_{j[i+1]}\vec{r}_j \\ &= P'_i(j_{|i})\vec{D}_{j[i+1]}\vec{r}_j = P'_i(j_{|i})\vec{u}_j \end{aligned}$$

In the above derivation, we first applied the definitions of P_{i+1} and $\vec{u}_j$, then we simplified the second term by noticing that $\vec{D}^\perp_{j[i+1]}\vec{D}_{j[i+1]} = 0$, then for the same exact reason we can add the component $P_j^{(j[i+1])}(j_{|i})\vec{D}_{j[i+1]}$, and finally we have the definition of P'_i.

Similarly, for the decryption oracle queries with input $\mathtt{C} = ([\vec{x}], [y]_T, \pi)$ where $\exists d : \vec{u} \in span(\vec{D}_d)$, we have that $P'_{i+1}(j_{|i+1})\vec{u} = P'_i(j_{|i})\vec{u}$. The derivation is identical as before. Thus, the two intermediate hybrids are equivalent.

Finally, we show that the second intermediate hybrid, $\mathbf{H}'_{i+1}$, is statistically close to $\mathbf{H}_{i,1,3}$, in fact, the only difference is that in the latter hybrid we use the function $P_{j+1}(\cdot)\vec{D}^\perp$. Equivalently as before, the two random functions are not equivalently distributed only when $span(\vec{D}_0 \| \vec{D}_1)$ has rank less than $2d$, which happens with probability at most $1/q$. Thus, we can conclude that $|\epsilon_{\mathbf{H}_{1,i,2}} - \epsilon_{\mathbf{H}_{1,i,3}}| \leq \frac{2}{q}$.

Hybrid $\mathbf{H}_{1,i,4}$. We remove the direct check $[\vec{u}]_1 \in span([\vec{D}_1]_1) \cup span([\vec{D}_0]_1)$ introduced in $\mathbf{H}_{1,i,2}$. This removal can only increase the winning probability of the adversary.

$$\epsilon_{\mathbf{H}_{1,i,3}} \leq \epsilon_{\mathbf{H}_{1,i,4}}.$$

Hybrid $\mathbf{H}_{1,i,5}$. To decrypt, we increase the number of keys used by the decryption oracle to compute the bit b_1.

$$\begin{aligned} &\mathcal{S} := \{j_{|i} \,\|b : b \in \{0,1\}\} && \text{if } \exists j \leq \mathsf{ctr} : \vec{D}^{*\perp}[\vec{x}] = \vec{D}^{*\perp}[\vec{x}_j] \\ &\mathcal{S} := \{j_{|i} \,\|b : b \in \{0,1\}, j \leq \mathsf{ctr}\} && \text{otherwise} \end{aligned}$$

This change can only increase the winning probability of the adversary since the set of the strings $\mathcal{S}$ used in $\mathbf{H}_{1,i,5}$ contains the set of strings used in $\mathbf{H}_{1,i,4}$.

As for non-critical queries, we need to show that the view of the adversary does not change: in particular, any non-critical query that decrypts to $\perp$ in $\mathbf{H}_{1,i,4}$ should decrypt to $\perp$ in $\mathbf{H}_{1,i,5}$ as well. This is easy to prove when the decryption query has $[\vec{u}] \in span([\vec{D}])$: indeed, even if we modify the set $\mathcal{S}$, this change does not affect the way we decrypt such queries (recall that any key $P_{i+1}(\cdot)$ is then multiplied by $\vec{D}^\perp$.) Also, a non-critical query could be a query for which

it holds that there exists $j \in [Q_{\mathsf{Enc}}]$ such that $\vec{D}^{*\perp}\vec{x}_j$ is equal to $\vec{D}^{*\perp}\vec{x}$. If a query of this form successfully decrypts in $\mathbf{H}_{1,i,4}$, the same happens in $\mathbf{H}_{1,i,5}$: again, this is because $\mathcal{S}$ in the latter hybrid is a superset of $\mathcal{S}$ in $\mathbf{H}_{1,i,4}$. But, it is still possible that a query of this form decrypts to $\perp$ in $\mathbf{H}_{1,i,4}$, but the "augmented" $\mathcal{S}$ in this new hybrid makes the consistency bit b_1 be 1, for some new key: we bound the probability of a similar event since we know that the only way to learn the image of the random function $P_{i+1}(\cdot)$ is via oracle queries to $\mathcal{O}_{\mathbf{dec}}$ and $\mathcal{O}_{\mathbf{enc}}$. By union bound, we obtain a statistical distance of $O(Q_{\mathsf{Enc}}Q_{\mathsf{Dec}}/q)$.

$$\epsilon_{\mathbf{H}_{1,i,4}} - O(Q_{\mathsf{Enc}}Q_{\mathsf{Dec}}/q) \leq \epsilon_{\mathbf{H}_{1,i,5}}.$$

Hybrid $\mathbf{H}_{1,i,6}$. This hybrid is equivalent to the previous one, but the decryption oracle computes a different set $\mathcal{S}$, as follows:

$$\begin{aligned} \mathcal{S} &:= \{j_{|i+1}\} && \text{if } \exists j \leq \mathsf{ctr} : \vec{D}^{*\perp}[\vec{x}] = \vec{D}^{*\perp}[\vec{x}_j] \\ \mathcal{S} &:= \{j_{|i+1} : j \leq \mathsf{ctr}\} && \text{otherwise} \end{aligned}$$

Notice that the set $\mathcal{S}$ as defined in $\mathbf{H}_{1,i,6}$ might be a (strict) subset of the set $\mathcal{S}$ as defined in $\mathbf{H}_{1,i,5}$. Thus, the distinguishing event is that the consistency check would pass in $\mathbf{H}_{1,i,5}$ but it would not pass in $\mathbf{H}_{1,i,6}$. In particular, such consistency check passes for an index of the form $j_i\|1$, such that $j[i+1] = 0$ and $j \leq \mathsf{ctr}$, and by the definition of the distinguishing event the integer representation of $(j_i\|1) \cdot 2^{|\log Q_{\mathsf{Enc}}|-i-1}$ is bigger than ctr. Thus, the key $\vec{f}^{\top} + P_i(j_i\|1)\vec{D}^{\perp}$ was never used for an encryption query. The only way an adversary can learn information about one of such keys is via decryption queries. In particular, each decryption query can at most decrease the set of possibilities (namely a valid y that matches the consistency check) by one. Moreover, the number of such keys is (very loosely) upper-bounded by Q_{Enc}, thus by union bound over all such keys and over all the decryption queries we obtain:

$$|\epsilon_{\mathbf{H}_{1,i,6}} - \epsilon_{\mathbf{H}_{1,i,5}}| \leq \frac{Q_{\mathsf{Enc}} \cdot Q_{\mathsf{Dec}}}{q - Q_{\mathsf{Dec}}}.$$

Hybrid $\mathbf{H}_{1,i+1,0}$. We then switch back the distribution of $[\vec{u}_j]$ to the span of $[\vec{D}_0]$. This transition is the reverse of what we have done to move from $\mathbf{H}_{1,i,0}$ to $\mathbf{H}_{1,i,1}$. We proceed in two steps:

- We first switch the j-th vector $[\vec{u}_j]$ computed by the encryption oracle to a vector in the span of $[(\vec{D}|\vec{U})]$, where $\vec{U}$ is uniform over $\mathbb{Z}_q^{n\times d}$, if the $(i+1)$-th bit of the binary representation of j is equal to 1.
- Then, we switch the j-th vector $[\vec{u}_j]$ computed by the encryption oracle to a vector in the span of $[\vec{D}_0]$.

Altogether we obtain and adversary $\mathcal{C}$ such that:

$$|\epsilon_{\mathbf{H}_{1,i+1,0}} - \epsilon_{\mathbf{H}_{1,i,6}}| \leq 2(n-d)\mathbf{Adv}^{\mathtt{mddh}}_{\mathbb{G}_1,\mathcal{D}_{n,d},\mathcal{C}}(\lambda) + \frac{2}{q-1}.$$

It is easy to see that $\epsilon_{\mathbf{H}_2} = \epsilon_{\mathbf{H}_{1,\lceil \log Q_{\mathsf{Enc}} \rceil,6}}$. Next, we prove that $\epsilon_{\mathbf{H}_2} \leq \frac{O(n^2)Q_{\mathsf{Enc}}Q_{\mathsf{Dec}}}{q}$. We reduce the adversary $\mathcal{A}$ playing in $\mathbf{H}_2$ to an (unbounded) adversary $\mathcal{B}$ upon which we can invoke the Lemma 7.3.1. We say that $\mathcal{B}$ *forged a valid tuple* if the output of $\mathcal{B}$ matches the event described in the lemma. For any assignments of the vector $\vec{a}$ and of the matrix $\vec{D}$ in the support of $\mathcal{D}_{n,d}$, we can consider in the Lemma 7.3.1 the matrix $\vec{E}$ to be set equal to $\vec{D}^*$.

Claim 7.6.1. $\Pr[\mathbf{H}_2 = 1] \leq \frac{O(n^2)Q_{\mathsf{Enc}}Q_{\mathsf{Dec}}}{q}$.

Let $(\vec{D}, \vec{D}^*, \vec{D}^\top \vec{f}, \vec{D}^\top \vec{F}, \vec{F}\vec{D}^*)$ be the tuple received by $\mathcal{B}$ from the challenger. The adversary $\mathcal{B}$ samples uniformly random values $(\vec{\vec{f}}, \vec{\vec{F}})$ such that $\vec{\vec{f}}^\top \vec{D} = \vec{f}^\top \vec{D}$, $\vec{\vec{F}}^\top \vec{D} = \vec{F}^\top \vec{D}$ and $\vec{\vec{F}}\vec{D}^* = \vec{F}\vec{D}^*$. We can think of the tuple $(\vec{\vec{f}}, \vec{\vec{F}})$ as a "fake" proving key that matches the verification key given by the challenger. Given $\vec{D}$ and $\vec{a}$, the reduction $\mathcal{B}$ can sample all the secret material needed to simulate the hybrid $\mathbf{H}_2$. In particular, it can compute the proving key and verification key of the proof system **PS** and sample the challenge bit. The reduction $\mathcal{B}$ samples an index value $j^*_{\mathsf{Enc}} \in [Q_{\mathsf{Enc}}]$ and an index $j^*_{\mathsf{Dec}} \in [Q_{\mathsf{Dec}}]$. Recall that Q_{Enc} and Q_{Dec} denote the number of encryption and decryption queries made by $\mathcal{A}$, respectively. At the j-th query to the encryption oracle:

- If $j \neq j^*_{\mathsf{Enc}}$, the reduction $\mathcal{B}$ generates $\vec{x}_j$ following the prescribed algorithms. Then, it computes $y_j \leftarrow \left(\left(\vec{\vec{f}} + \vec{\vec{F}}\vec{x}_j\right)^\top + P(j)\vec{D}^\perp\right)\vec{u}_j$, where we recall that $P(\cdot)$ is a random function.
- Else, for $j = j^*_{\mathsf{Enc}}$, $\mathcal{B}$ computes $\vec{x}_j$ as prescribed, queries its own oracle with $\vec{x}_j$ and obtains a value $\vec{v} = \vec{f} + \vec{F} \cdot \vec{x}_j$, then, it uses $\vec{v} + P(j)\vec{D}^\perp$ to compute the proof y, associated with $\vec{u}_j$, namely: $y_j \leftarrow \left(\vec{v}^\top + P(j)\vec{D}^\perp\right)\vec{u}_j$.

At the j-th query to decryption oracle with ciphertext $\mathtt{C} = ([\vec{x}], [y]_T, \pi)$ there are three possible cases. The easiest case to handle is if $\vec{u} \in span(\vec{D})$ or $\exists j \neq j^*_{\mathsf{Enc}}$ such that $\vec{D}^{*\perp}\vec{x}_j = \vec{D}^{*\perp}\vec{x}$. The reduction $\mathcal{B}$ can compute the consistency check using the keys $\vec{\vec{f}}, \vec{\vec{F}}$ and the random function P.

The second case is when $\vec{D}^{*\perp}\vec{x}_{j^*_{\mathsf{Enc}}} = \vec{D}^{*\perp}\vec{x}$, in this case let $\vec{r}'$ be such that $\vec{x} - \vec{x}_{j^*_{\mathsf{Enc}}} = \vec{D}^*\vec{r}'$ and compute

$$y' \leftarrow y_{j^*_{\mathsf{Enc}}} + \vec{f}^\top \vec{D}\vec{r}' + \vec{x}^\top_{j^*_{\mathsf{Enc}}} \vec{F}^\top \vec{D}\vec{r}' + (\vec{F}\vec{D}^*\vec{r}')^\top(\vec{u}_{j^*_{\mathsf{Enc}}} + \vec{D}\vec{r}')$$

namely, compute the element $[y']_T$ as if it was computed in the re-randomization of the ciphertext $\mathtt{C}_{j^*_{\mathsf{Enc}}}$ using randomness $\vec{r}'$. Notice that, by definition of $\mathbf{H}_2$ the consistency check for $[y]_T$ would be computed by checking if

$$y \stackrel{?}{=} \left(\left(\vec{f} + \vec{F}\vec{x}\right)^\top + P(j^*_{\mathsf{Enc}})\vec{D}^\perp\right)\vec{u}.$$

By Lemma 7.6.1 and by definition of $y_{j^*_{\mathsf{Enc}}}$, the two checks are equivalent. The last case is when $\vec{u} \notin span(\vec{D}) \wedge \forall j : \vec{D^*}^\perp \vec{x}_j \neq \vec{D^*}^\perp \vec{x}$, i.e., the query might be "critical":

- If $j < j^*_{\mathsf{Dec}}$ then return $\perp$ to the adversary $\mathcal{A}$, in this case we assume that the query was not critical and that the decryption would fail.

- If $j = j^*_{\mathsf{Dec}}$ then output the tuple $(y - P(j^*_{\mathsf{Enc}})\vec{D}^{\perp}\vec{u}, \vec{u}, \vec{x})$ as the forgery of $\mathcal{B}$.

We condition on the event that j^*_{Dec} is the first critical query of $\mathcal{A}$ and that, let the ciphertext sent by $\mathcal{A}$ at the j^*_{Dec} query be $\mathtt{C} = ([\vec{x}], [y]_T, \pi)$ we have that the equation $[y]_T = (\vec{f} + P(j^*_{\mathsf{Enc}})\vec{D}^{\perp} + \vec{F}\vec{x})^{\top}[\vec{u}]$ holds. Let $\mathtt{Guess}$ be such event. Conditioned on such a lucky event, the adversary $\mathcal{B}$ indeed produces a valid forgery, in fact by the definition of a critical query $(\vec{x}_{j^*_{\mathsf{Enc}}} - \vec{x}) \notin span(\vec{D}^*)$ and $\vec{u} \notin span(\vec{D})$.

We show that the view provided by $\mathcal{B}$ to the adversary $\mathcal{A}$ up to the j^*_{Dec}-th decryption query and conditioned on $\mathtt{Guess}$ is equivalent to the view of the adversary up to the j^*_{Dec}-th decryption query in the hybrid game $\mathbf{H}_2$. The intuition is that the values $P(j)\vec{D}^{\perp}$, for all j, mask the components of $(\vec{f}, \vec{F})$ and $(\vec{\bar{f}}, \vec{\bar{F}})$ that differ. Indeed, we know that for some row vectors $\vec{v}, \vec{w}, \vec{w}'$, it holds that $\vec{f} = \vec{D}\vec{v} + (\vec{w}\vec{D}^{\perp})^{\top}$ and $\vec{\bar{f}} = \vec{D}\vec{v} + (\vec{w}'\vec{D}^{\perp})^{\top}$. Similarly, for some $\vec{V}, \vec{W}$ and $\vec{W}'$, $\vec{F} = \vec{D}\vec{V} + (\vec{W}\vec{D}^{\perp})^{\top}$, and $\vec{\bar{F}} = \vec{D}\vec{V} + (\vec{W}'\vec{D}^{\perp})^{\top}$.

Let P' be a uniformly random function, and consider the following function:

$$P(j) = \begin{cases} P'(j), & j = j^*_{\mathsf{Enc}} \\ P'(j) + \Delta_j, & j \neq j^*_{\mathsf{Enc}} \end{cases}$$

where $\Delta_j = \vec{w} - \vec{w}' + \vec{x}_j^{\top}(\vec{W} - \vec{W}')$. It is not hard to see that P is a uniformly random function. Now consider the mental experiment where $\mathcal{B}$ runs the same but using the random function P defined above. Since P is uniformly random, the probability that $\mathcal{B}$ forges a valid tuple in this mental experiment is the same as the probability that $\mathcal{B}$ forges a valid tuple in the real experiment. Also, for any $j \neq j^*_{\mathsf{Enc}}$ the value y computed at the j-th encryption oracle query is:

$$\begin{aligned} y &= \left(\left(\vec{\bar{f}} + \vec{\bar{F}}\vec{x}_j\right)^{\top} + P(j)\vec{D}^{\perp}\right)[\vec{u}_j] = \left(\left(\vec{\bar{f}} + \vec{\bar{F}}\vec{x}_j\right)^{\top} + (P'(j) + \Delta_j)\vec{D}^{\perp}\right)[\vec{u}_j] = \\ &= \left(\left(\vec{\bar{f}} + ((\vec{w} - \vec{w}')\vec{D}^{\perp})^{\top} + (\vec{\bar{F}} + ((\vec{W} - \vec{W}')\vec{D}^{\perp})^{\top})\vec{x}_j\right)^{\top} + P'(j)\vec{D}^{\perp}\right)[\vec{u}_j] = \\ &= \left(\left(\vec{f} + \vec{F}\vec{x}_j\right)^{\top} + P'(j)\vec{D}^{\perp}\right)[\vec{u}_j]. \end{aligned}$$

The probability that the reduction $\mathcal{B}$ creates a forgery is $\Pr[\mathbf{H}_2 = 1 \wedge \mathtt{Guess}]$, and the two events are independent. Moreover, since $\Pr[\mathtt{Guess}] = (Q_{\mathsf{Enc}}Q_{\mathsf{Dec}})^{-1}$, by Lemma 7.3.1 we have that $\Pr[\mathbf{H}_2 = 1] \leq \frac{n(n+1)Q_{\mathsf{Enc}}Q_{\mathsf{Dec}}}{q}$. □

7.6.1 Publicly-Verifiable Rand-RCCA PKE

We show two publicly verifiable Rand-RCCA PKE schemes based on the scheme from Section 7.6. Following the ideas in [FFHR19], we append a malleable NIZK proof (essentially a Groth-Sahai proof) that $[y]_T$ and π are well-formed to the ciphertexts of PKE from the previous section. The decryption algorithm outputs the decrypted message only if the NIZK proofs are valid. Public verifiability follows because the NIZK proofs can be verified using the public parameters.

Let $\mathsf{PKE}_1 = (\mathsf{KGen}_1, \mathsf{Enc}_1, \mathsf{Dec}_1, \mathsf{Rand}_1)$ be the scheme of Section 7.6 instantiated using the benign proof system of Section 7.4.1, and let PEvl_2 and $\mathbf{PS}_2 = (\mathsf{PGen}_2, \mathsf{PPrv}_2, \mathsf{PVer}_2)$ form a

malleable NIZK system for membership in the relation

$$\mathcal{R}_2 = \left\{ (\mathsf{pk}, [\vec{x}]), ([y]_T, \pi, \vec{r}) : \begin{array}{c} y = \vec{f}^\top \vec{u} + \vec{x}^\top \vec{F} \vec{u} \\ \mathsf{PPrv}_1(ppk, [\vec{u}], \vec{r}) = \pi \end{array} \right\},$$

and where the allowable set of transformations contains all the transformations $(T_{\mathsf{el}}, T_{\mathsf{wit}})$ such that it exists $\vec{\hat{r}}$ with $T_{\mathsf{el}}(\mathsf{pk}, [\vec{x}]) = \mathsf{pk}, [\vec{\hat{x}}]$, $T_{\mathsf{wit}}([y]_T, \pi, \vec{r}) = [\hat{y}]_T, \hat{\mathsf{pk}}, \vec{r} + \vec{\hat{r}}$ and $([\vec{\hat{x}}], [\hat{y}]_T, \hat{\pi}) = \mathsf{Rand}_1(\mathsf{pk}, ([\vec{x}], [y]_T, \pi); \vec{\hat{r}})$; each transformation in the set of allowable transformation is uniquely identified by a vector $\vec{\hat{r}}$.

The pv-Rand-PKE scheme $\mathsf{PKE}_2 = (\mathsf{Init}, \mathsf{KGen}_2, \mathsf{Enc}_2, \mathsf{Dec}_2, \mathsf{Rand}_2, \mathsf{Ver})$ is identical to PKE_1, except that

- KGen_2 additionally samples the common reference string for $\mathbf{PS}_2$,
- the encryption procedure computes a ciphertext as in PKE_1 but additionally computes a proof π_2 for $\mathbf{PS}_2$ and outputs a ciphertext $\mathtt{C} = ([\vec{x}_1], \pi_2)$,
- the decryption procedure first checks the proof π_2 holds w.r.t. the instance $(\mathsf{pk}, [\vec{x}])$ and, if so, it outputs $\mathtt{msg} = (-\vec{a}^\top, 1)[\vec{x}]$ (and $\perp$ otherwise),
- the re-randomization procedure randomizes $[\vec{x}]$ as in PKE_1 and uses PEvl_2 for the remaining part of the ciphertext, and
- Ver_2 simply checks the proof π_2.

Theorem 7.6.2. *If $\mathbf{PS}_2$ is adaptively sound, $(\epsilon, O(T))$-composable zero-knowledge, and perfect derivation private, and PKE_1 is mRCCA secure then PKE_2 is publicly verifiable, perfectly re-randomizable, and mRCCA-secure. Specifically, for any PPT $\mathcal{A}$ making up to Q_{Enc} encryption queries and Q_{Dec} decryption queries and with running time T exist PPT $\mathcal{B}^{\mathrm{rcca}}$ making the same number of queries and adversaries $\mathcal{B}^{\mathrm{snd}}, \mathcal{B}^{\mathrm{zk}}$ with similar running times*

$$\mathbf{Adv}^{\mathtt{mRCCA}}_{\mathcal{A},\mathsf{PKE}_2}(\lambda) \leq \mathbf{Adv}^{\mathtt{mRCCA}}_{\mathcal{B}^{\mathrm{rcca}},\mathsf{PKE}_1}(\lambda) + \mathbf{Adv}^{\mathtt{snd}}_{\mathcal{B}^{\mathrm{snd}},\mathbf{PS}_2}(\lambda) + \epsilon$$

The proof of the theorem follows by inspection of the proof of Theorem 2 in [FFHR19]. In more detail, their proof proceeds in two steps. First, it reduces to the adaptive soundness of the NIZK proof system to claim that if a *publicly-verifiable* ciphertext decrypts correctly then its respective *non-publicly verifiable* ciphertext should decrypt correctly too. We notice that this step can be performed tightly relying either on statistical adaptive soundness of the proof system or relying on the computational soundness of the proof system when the language proved by the proof system is witness samplable. The reason is that the reduction can check which one of the many NIZK-proofs from the adversary breaks adaptive soundness before submitting it as its forgery. The second step uses composable zero-knowledge to first tightly switch the way the public parameters are generated and then to switch (all together) the proofs for the ciphertexts from real to simulated.

To instantiate the malleable NIZK, we consider a construction along the same line of [FFHR19]. In more detail, [FFHR19] introduced an extension of the Groth-Sahai proof system that is zero-knowledge even for pairing product equations where the $\mathbb{G}_T$-elements are variables. Their idea is to commit the elements in $\mathbb{G}_T$ using a commitment scheme with nice bilinear properties. Groth-Sahai Proofs can be instantiated under any $\mathcal{D}_k$-MDDH Assumption [EHK+13]

and, given their nice algebraic properties they posses malleability property [CKLM12]. The verification equation uses a special projecting bilinear map $\tilde{e} : \mathbb{G}_1^{k+1} \times \mathbb{G}_2^{k+1} \to \mathbb{G}_T^m$. In general, the map $\tilde{e}$ with the optimal m depends on $\mathcal{D}_k$ (not only on k), as was proven [HHH+14]. Groth-Sahai Proofs are perfect adaptive sound, however we also make use of a QA-NIZK for linear spaces [KW15] which is computationally adaptive sound. Fortunately, the language to be proved is witness sampleable, thus the tight-security of PKE_2 holds. Recall that we need a NIZK for the following relationship:

$$\mathcal{R}_2 = \left\{ (\mathsf{pk}, [\vec{x}]_1), ([y]_T, \pi, \vec{r}) : \begin{array}{c} y = \vec{f}^\top \vec{u} + \vec{x}^\top \vec{F} \vec{u} \\ \mathsf{PPrv}_1(ppk, [\vec{u}]_1, \vec{r}) = \pi \end{array} \right\},$$

We follow a similar approach to [FFHR19], specifically the proof includes:

1. Commitments to $[y]_T$ and commitments to $[\vec{u} \otimes \vec{u}]_T$.
2. A commitment $[\vec{c}_0]_1 \in \mathbb{G}_1^3$ to $[\vec{f}^\top \vec{D} \vec{r}]_1$ and commitments $[\vec{c}_1]_1, \ldots, [\vec{c}_{n+1}]_1$ to the components of the vector $[\vec{F}^\top \vec{D}^* \vec{r}]$.
3. Commitments $[\vec{c}_{n+2}]_1, \ldots, [\vec{c}_{n+2+nd}]_1$ to the elements of the vector $[\vec{u} \otimes \vec{r}]_1$.
4. An (extended) GS proof of the equation $[y]_T = [1]_1 \cdot [\vec{f}^\top \vec{D} \vec{r}]_1 + [\vec{x}^\top]_1 \cdot [\vec{F}^\top \vec{D} \vec{r}]_1$.
5. An (extended) GS proof of the equation $[\vec{k}^\top \vec{D} \otimes \vec{I}]_1 \cdot [\vec{u} \otimes \vec{r}]_1 = \pi$.
6. n^2 (extended) GS proofs of the equations $[\vec{I} \otimes \vec{D}]_1 \cdot [\vec{u} \otimes \vec{r}]_1 = [\vec{u} \otimes \vec{u}]_T$.
7. A proof that the commitments $[\vec{c}_0]_1, \ldots, [\vec{c}_{n+1}]_1$ are well-formed. This can be proven with one proof of membership in a linear space in each group [JR14, KW15]. With more detail, the proof shows that $[\vec{u}]_1 = [\vec{D}]_1 \vec{r}$ and that the commitments commit to $\vec{r}$ using the basis $[\vec{f}^\top \vec{D}]_1$ and $[\vec{F}^\top \vec{D}]_1$ to commit.

As the reader might have noticed, the proof is quite expensive, as it contains more than mn^2 elements in $\mathbb{G}_T$. In the next paragraph we show how to reduce the size of the proof relying on a stronger cryptographic assumption.

A more efficient tight-secure pv-Rand-RCCA PKE. To facilitate our more efficient scheme, we introduce a stronger variant of the MDDH assumption (cf. Definition 2.3.5) in which the adversary gets not only a matrix $[\vec{A}]$, but also the tensor product $[\vec{A} \otimes \vec{A}]$ in order to distinguish an element from $span([\vec{A}])$ and random.

Definition 7.6.1 (Tensor Matrix Diffie-Hellman assumption in $\mathbb{G}_\gamma$)**.** *The $\mathcal{D}_{\ell,k}$-Tensor-Matrix-Decisional-Diffie-Hellman (TMDDH) assumption in group $\mathbb{G}_\gamma$ holds if for all non-uniform PPT adversaries $\mathcal{A}$,*

$$\left| \Pr\left[\mathcal{A}(\mathcal{G}, [\vec{A} \otimes \vec{A}]_\gamma, [\vec{A}]_\gamma, [\vec{A}\vec{w}]_\gamma) = 1 \right] - \Pr\left[\mathcal{A}(\mathcal{G}, [\vec{A} \otimes \vec{A}]_\gamma, [\vec{A}]_\gamma, [\vec{z}]_\gamma) = 1 \right] \right|$$

is negligible, where the probability is taken over $\mathcal{G} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathcal{P}_1, \mathcal{P}_2) \leftarrow \mathsf{GroupGen}(1^\lambda)$, $\vec{A} \leftarrow \mathcal{D}_{\ell,k}, \vec{w} \leftarrow \mathbb{Z}_q^k$, $[\vec{z}]_\gamma \leftarrow \mathbb{G}_\gamma^\ell$, and the coin tosses of adversary $\mathcal{A}$.

The TMDDH assumption can be seen as a generalization of the "square-Diffie-Hellman" assumption [BDS98, MW96], and as a special case of the "Uber assumption family" [Boy08]. Since a TMDDH adversary gets quadratic terms $[\vec{A} \otimes \vec{A}]$ "in the exponent", it is not clear how this assumption relates to the more standard MDDH assumption. However, we remark that the TMDDH assumption holds generically for large enough dimensions, at least for uniformly random $\vec{A}$.

To explain what we mean by "holds generically", we give a brief explanation of the generic group model (in the formulation of Maurer [Mau05]). A *generic adversary* on assumptions in a group only interacts through an oracle with the investigated group. Concretely, this adversary initially gets only the group order and so-called *handles* (i.e., running numbers) to the challenge group elements. The adversary additionally gets access to oracles that allow to test group elements for zero (which also allows testing for equality), perform the group operation on two handles (which yields a new handle to the resulting group element), and potentially to other operations involving group elements (such as a pairing).

It is possible to show that relative to generic adversaries, the Diffie-Hellman and discrete logarithm assumptions hold [Nec94, Sho97b]. In fact, the $\mathcal{U}_{k+1,k}$-MDDH assumption even holds against generic adversaries with a symmetric k-linear map [EHK+13]. We remark that this last fact implies that also the $\mathcal{U}_{k+1,k}$-TMDDH assumption holds generically, even in the presence of a (symmetric) pairing.

Lemma 7.6.3 (Generic security of TMDDH)**.** *For $k \geq 4$, the $\mathcal{U}_{k+1,k}$-TMDDH assumption holds against generic adversaries in a symmetric pairing setting.*

The idea of the second publicly-verifiable PKE scheme is to (1) add in the public key the values $\vec{k}^\top[\vec{D} \otimes \vec{D}]$ and (2) use a malleable proof system $\mathbf{PS}_3$ for membership in the relation

$$\mathcal{R}_3 = \left\{ (\mathsf{pk}, [\vec{x}]), ([y]_T, \pi, \vec{r}) : \begin{array}{c} y = \vec{f}^\top \vec{u} + \vec{x}^\top \vec{F} \vec{u} \\ \vec{k}^\top [\vec{D} \otimes \vec{D}] \vec{r} \otimes \vec{r} \cdot [1] = \pi \end{array} \right\},$$

with the same set of allowable transformation as in the previous publicly verifiable PKE scheme. The languages associated to the relation $\mathcal{R}_2$ and $\mathcal{R}_3$ are identical, but we can obtain a more efficient NIZK proof for the relation $\mathcal{R}_3$. More in detail, in contrast with the NIZK-proof of PKE_2, the NIZK proof of PKE_3:

- It does not include the commitments to $[\vec{u} \otimes \vec{r}]$, but instead it includes commitments $[\vec{c}_{n+2}], \ldots, [\vec{c}_{n+2+d^2}]$ to the components of the vector $\vec{r} \otimes \vec{r}$.
- It does not contain the (extended) GS proofs of step 5 and 6, but instead it contains a (standard) GS proof for the equations $\vec{k}^\top[\vec{D} \otimes \vec{D}](\vec{r} \otimes \vec{r}) \cdot [1] = \pi$.

Theorem 7.6.3. *The pv-Rand-PKE scheme PKE_3 is publicly verifiable, perfectly re-randomizable and RCCA-secure. Specifically:*

$$\begin{aligned} \mathbf{Adv}^{\mathtt{RCCA}}_{\mathcal{A},\mathsf{PKE}}(\lambda) \leq & \mathbf{Adv}^{\mathtt{TMDDH}}_{\mathbb{G}_1,\mathcal{U}_{n,d},\mathcal{B}}(\lambda) + O\left(d \log Q_{\mathsf{Enc}}\right) \cdot \mathbf{Adv}^{\mathtt{MDDH}}_{\mathbb{G}_1,\mathcal{U}_{n,d},\mathcal{B}'}(\lambda) \\ & + \log Q_{\mathsf{Enc}} \cdot \mathbf{Adv}^{\mathtt{snd}}_{\mathcal{B}'',\mathbf{PS}}(\lambda) + O\left(\tfrac{n^2 Q_{\mathsf{Dec}} Q_{\mathsf{Enc}} \log Q_{\mathsf{Enc}}}{q}\right) \end{aligned}$$

We only sketch the proof, which is only a slight variation of the proof of Theorem 7.6.1. Notice that in the proof of Theorem 7.6.1 to move from $\mathbf{G}_3$ to $\mathbf{G}_4$ we use the $\mathcal{D}_{n,d}$-MDDH

assumption. This step changes with our modified scheme, since we add $[\vec{D} \otimes \vec{D}]$ to the public key. We thus need to rely on the stronger TMDDH assumption. Also notice that this is the only step in the proof of Theorem 7.6.1 where the assumption over the matrix $[\vec{D}]$ is used. Finally, observe that we can prove both composable zero-knowledge and computational adaptive soundness of the NIZK proof system for $\mathcal{R}_3$ using the classical $\mathcal{D}_k$-MDDH assumption.

7.7 Application: Universally Composable MixNet

We can plug-and-play our pv-Rand-RCCA PKE schemes in the MixNet protocol of [FFHR19] because their protocol works for any pv-Rand RCCA scheme that has the property of being *linear* and a property that holds for both PKE_2 and PKE_3.

7.7.1 Linear pv-Rand PKE

Definition 7.7.1 (Linear pv-Rand-RCCA PKE,[FFHR19])**.** *We say that a pv-Rand-RCCA PKE scheme is* linear *if there exist a group $\mathbb{G}$ (for example $\mathbb{G} = \mathbb{G}_1$) and parameters $\ell, \ell', \ell'' \in \mathbb{N}$ such that (1) every key pair $(\mathsf{pk}, \mathsf{sk})$ we can parse $\mathsf{pk} = ([\vec{P}], \hat{\mathsf{pk}})$ and $\mathsf{sk} = (\vec{S}, \hat{\mathsf{sk}})$, where $[\vec{P}] \in \mathbb{G}^{\ell \times \ell''}$ and $\vec{S} \in \mathbb{Z}_q^{\ell' \times \ell}$, (2) any ciphertext $\mathtt{C} \in \mathcal{C}$ can be parsed as $([\vec{y}], \hat{\mathtt{C}})$ where $[\vec{y}] \in \mathbb{G}^{\ell}$, (3) for any ciphertext $\mathtt{C}$ such that $\mathsf{Ver}(\mathsf{pk}, \mathtt{C}) = 1$ the decryption procedure is linear, i.e., we have $\mathsf{Dec}(\mathsf{sk}, \mathtt{C}) = \vec{S} \cdot [\vec{y}]$ (4) let $\mathtt{C}' = \mathsf{Rand}(\mathsf{pk}, \mathtt{C}; \vec{r}, r)$ where $\mathtt{C}' = ([\vec{y}'], \hat{\mathtt{C}}')$ be a re-randomization of $\mathtt{C} = ([\vec{y}], \hat{\mathtt{C}})$ and $\vec{r} \in \mathbb{Z}_q^{\ell''}$ then $([\vec{y}] - [\vec{y}']) = [\vec{P}]\vec{r}$.*

Lemma 7.7.1. *The pv-Rand-RCCA PKE schemes PKE_2 and PKE_3 defined in Section 7.6.1 are linear schemes.*

Proof. Given the definition of $[\vec{P}]_1, \vec{S}, \vec{y}, l, l', l''$ in Definition 7.7.1, we set for both schemes the following:

- $l := n + 1$, $l' := 1$ and $l'' := d$
- $\vec{P} := \vec{D}^*$
- $\vec{S} := (-\vec{a}^\top, 1)$
- $\vec{y} := \vec{x}$

Property 1 and 2 of Definition 7.7.1 easily follow from the above definitions. Property 3 holds because if the verification procedure outputs 1 when given in input the ciphertext $\mathtt{C}$, the decryption algorithm returns the message $[p]_1 - [\vec{a}^\top \vec{u}]_1 = (-\vec{a}^\top, 1)[\vec{x}]_1 = \vec{S} \cdot \vec{y}$. Property 4 can be easily proved since the difference between $[\vec{\hat{x}}]_1$ and $[\vec{x}]_1$, where $[\vec{\hat{x}}]_1$ is the first component of $\mathsf{Rand}(psk, \mathtt{C}, \vec{\hat{r}})$, is equal to $[\vec{D}^*]_1 \vec{\hat{r}} = [\vec{P}]_1 \vec{\hat{r}}$. □

We notice that to obtain our "tightly-secure" MixNet we need only to make sure that the Rand-mRCCA PKE and the simulation-extractable NIZK proofs are tight secure. Let $\mathbf{Adv}^{\mathtt{sim-ext}}_{\mathcal{A},\mathbf{PS}}(\lambda)$ be the advantage of an adversary $\mathcal{A}$ against the simulation extractability experiment for $\mathbf{PS}$, we are ready now to state the main contribution of this section.

Theorem 7.7.1. *Let* PKE *be a linear pv-Rand RCCA PKE,* $\mathbf{PS}$ *be a simulation-extractable NIZK, and let* Π *be the MixNet protocol from [FFHR19] instantiated with* PKE *and* $\mathbf{PS}$*. The protocol* Π $\mathsf{realizes}$ $\mathcal{F}_{\mathsf{Mix}}$ *with setup assumptions a threshold decryption functionality* $\mathcal{F}_{\mathsf{TDec}}[\mathsf{PKE}]$ *and a common-reference string functionality* $\mathcal{F}_{\mathsf{CRS}}$*. More in detail, there exist a simulator* $\mathcal{S}$ *and negligible function* $\mathsf{negl}(\lambda, m)$ *such that for any static-corruption environment* $\mathcal{Z}$ *with running time* $T_{\mathcal{Z}}$ *there exist adversaries* $\mathcal{B}, \mathcal{B}'$ *whose running time is* $O(T_{\mathcal{Z}}(\lambda))$*, such that:*

$$\Big|\Pr[REAL_{\mathcal{Z},\Pi}(\lambda)=1] - \Pr\Big[\{\mathcal{F}_{\mathsf{CRS}},\mathcal{F}_{\mathsf{TDec}}\}\text{-}HYBRID^{\mathcal{F}_{\mathsf{Mix}}}_{\mathcal{Z},\mathcal{S}}(\lambda)=1\Big]\Big| \\ \leq 3\mathbf{Adv}^{\mathtt{mRCCA}}_{\mathcal{B},\mathsf{PKE}}(\lambda) + \mathbf{Adv}^{\mathtt{sim-ext}}_{\mathcal{B}',\mathbf{PS}'}(\lambda) + \mathsf{negl}(\lambda, m)$$

We stress that the negligible function $\mathsf{negl}(\lambda, m)$ in the statement of Theorem 7.7.1 is independent of the running time of the environment $T_{\mathcal{Z}}$, and only depends on the number of mixers of the protocol (which we can think as a small number). The proof of the theorem follows by inspection of the proof of Theorem 5 in [FFHR19] and observing that the three steps of the proof that reduce to the pv-Rand-RCCA security of PKE can be performed tightly by relying on the multi-ciphertext RCCA security definition (cf. Definition 7.5.1). In Section 7.7.2 we give more details, and we show how to instantiate the necessary simulation-extractable NIZK using the tightly-secure QA-NIZK based on the MDDH assumption of Abe *et al.* [AJOR18]. Thus, instantiating the protocol with PKE_2 (resp. PKE_3) we obtain a MixNet protocol that reduces almost-tightly in the number of mixed messages to the MDDH (resp. TMDDH) Assumption.

7.7.2 Quasi-Adaptive NIZK for the Input-Submission Phase

The MixNet protocol, when instantiated with either PKE_2 or PKE_3 also needs a simulation-extractable NIZK proof system for the relation:

$$\mathcal{R}' = \{(\mathsf{pk}, \mathtt{C}), (\vec{r}, \mathtt{msg}) : [\vec{x}]_1 = [\vec{D}^*]\vec{r} + \vec{e}\mathtt{msg}\}$$

where $\vec{e}$ is the $(n+1)$-th basis. In particular, it is sufficient the notion of simulation f-extractability, where f is the efficiently computable function that maps tuples $(\mathtt{msg}, \vec{r})$ to $\mathtt{msg}$. Moreover, we can focus on quasi-adaptive NIZK that works for distribution of relation indexed by the public key pk.

We show a simulation f-extractable QA-NIZK based on the simulation-sound QA-NIZK for linear spaces of Abe *et al.* [AJOR18].

Let $\mathbf{PS} = (\mathsf{PGen}, \mathsf{PPrv}, \mathsf{PVer})$ be a simulation-sound QA-NIZK for linear spaces. Consider the proof system $\mathbf{PS}' = (\mathsf{PGen}', \mathsf{PPrv}', \mathsf{PVer}')$ for the relation is $\mathcal{R}'_{\mathsf{pk}} = \{[\vec{x}], (\mathtt{msg}, \vec{r}) : [\vec{x}] = [\vec{D}^*]\vec{r} + \vec{e}\mathtt{msg}\}$ of $\mathbb{Z}_q^{n+1}$ and $\vec{D}^*$ is sampled as described in the description of the RCCA-secure PKE of Section 7.6 and where:

- $\mathsf{PGen}'(1^\lambda)$ samples $\vec{U} \leftarrow\!\!\$\ \mathcal{U}_{n+1,2}$ runs $\mathsf{crs} \leftarrow\!\!\$\ \mathsf{PGen}(1^\lambda, [\vec{P}]_1)$ where $\vec{P}$ is the matrix $\vec{D}^* \| \vec{U}$ and outputs $\mathsf{crs}' = (\mathsf{crs}, [\vec{U}]_1)$.
- $\mathsf{PPrv}([\vec{x}], (\mathtt{msg}, \vec{r}))$ samples $\vec{s}$, computes $[\vec{c}] = [\vec{U}]\vec{s} + \vec{e}\mathtt{msg}$, and then it computes a proof π using PPrv that $([\vec{c}] - [\vec{x}]) \in span([\vec{P}]_1)$ (whose witness is the vector $(\vec{r}^\top, \vec{s}^\top)^\top$) and it outputs $\pi, [\vec{c}]$.

- $\mathsf{PVer}([\vec{x}], \pi')$ verifies that $[\vec{c} - \vec{x}] \neq [\vec{0}]$ and that π is a valid proof of membership in the linear space spanned by $[\vec{P}]_1$ for $[\vec{c} - \vec{x}]_1$.

Lemma 7.7.2. *Let f be the function that maps $(\mathtt{msg}, \vec{r})$ to $\mathtt{msg}$, For any adversary $\mathcal{A}$ exists adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\mathtt{sim-ext}}_{\mathcal{A},\mathbf{PS}',f}(\lambda) \leq \mathbf{Adv}^{\mathtt{sim-sound}}_{\mathcal{B},\mathbf{PS}}(\lambda) + O(1/q).$$

The extractor uses the trapdoor $\vec{U}$ to extract from $[\vec{c}]$ the message $\mathtt{msg}$. Assume that the proof verifies but that the extracted message $\mathtt{msg}'$ is not the same as the message $\mathtt{msg}$ encrypted in $\vec{x}$, then $\vec{x} - \vec{c} \in \mathit{span}(\vec{P})$ implies that $\vec{e} \in \mathit{span}(\vec{P})$ which can happen with probability smaller than $O(1/q)$ over the choice of $\vec{U}$ and $\vec{D}^*$.

Chapter 8

Conclusion

We presented in this thesis several results that deal with different aspects of the malleability and non-malleability of cryptographic protocols, and our results aimed at improving the understanding of the interplay and the relationship between these two properties.

However, there are still questions that remain open and in the next section we state some of them.

8.1 Open problems

8.1.1 Non-malleability of zkSNARKs

Our framework in Chapter 3 is general enough to handle compilation from polynomial commitment schemes different from KZG. Our contribution identifies a set of properties that a polynomial commitment scheme needs to have so that the resulting SNARK is simulation-extractable. We believe that, thanks to the non malleability of random oracles, the FRI polynomial commitment scheme [BGKS20] readily possesses the necessary properties, which would imply the simulation extractability of STARKs [BBHR19].

We leave as an open problem to extend our result to even more flexible polynomial IOP models.

Recent works extend the polynomial evaluation proofs of KZG to multiple evaluation points [TAB+20, ZBK+22]. Our simulation extractability strategy for KZG can be applied partially to these schemes; however, our technique uses a clever argument to separate the realm of commitments from the realm of proofs (in KZG proofs and commitments are both of the form $[p(s)]_1$ for some polynomial p) based on their degree as polynomials. Unfortunately, the same technique does not work when the degree of the polynomial in the proof depends on the number of evaluation points in the proved statement.

UC-secure SNARKs. An interesting line of research tries to address the problem of obtaining UC-secure SNARKs, which captures a form of non-malleability. The work of Abdolmaleki, Ramacher and Slamanig [ARS20] shows a generic compiler to simulation-extractable SNARKs which requires key-homomorphic signatures. Their compiler produces universally-composable SNARKs (UC-SNARKs), which they prove through black-box straight-line extractor. To obtain a black-box straight-line extractor, they append to the SNARK proof an encryption of the witness, thus achieving a relaxed succinctness w.r.t. the size of the circuit describing the

relation. The work of Ganesh, Kondi, Orlandi, Pancholi and Takahashi [GKO+23] shows how to regain full succinctness in UC-SNARKs in the ROM through Fischlin's transform [Fis05]. The work of [CF24] shows that a wide class of zkSNARKs is UC-secure when modeling the Random Oracle as a shared and global functionality, although it is *programmable* in a restricted way. In [BCC+24] we show that transparent UC-SNARKs can be obtained even without the need for a programmable RO at the cost of slightly weakening the UC model.

8.1.2 Composition of CP zkSNARKs and non-malleability

We foresee applications for our toolbox results presented in Chapter 5 beyond zkVMs. For example, it could be used to provide alternative proofs for the simulation extractability of Spartan and Bulletproofs potentially substantially simplifying the approach in [DG23] and our own approach for zkLasso with it. Spartan in particular is a good candidate for this given its several moving parts which can be seen as separate block (the Hyrax polynomial commitment, grand product arguments, etc.).

8.1.3 Tight security of Rand-RCCA PKE schemes

Our approach is semi-generic, as we work with pairing-based cryptography. We leave as open problem to provide a generic framework to instantiate (almost) tightly-secure Rand-RCCA-secure PKE. Possible starting points are the HPS-based frameworks of [WCY+21] for Rand-RCCA schemes and [HLLG19] for tightly-secure (LR-)CCA-secure schemes.

We leave as open problem the extension of our analysis to tightly-secure leakage-resilient RCCA PKE schemes to give an efficient instantiation based on non publicly-verifiable Rand-RCCA PKE schemes and apply the benefits of the mix-net protocol described in Chapter 6.

8.1.3.1 On the Instantiability with Asymmetric Pairings

Our construction requires type-1 pairings, which are less efficient than type-3 ones. Hence, it is natural to ask whether we can instantiate our construction also from type-3 pairings. Unfortunately, we do not know how to do so, since it is not clear how to reconcile the adaptive partitioning technique [Hof17] with a Rand-RCCA construction in settings with type-3 pairings (such as the one from [FFHR19]).

In a nutshell, ciphertexts from the Rand-RCCA construction from [FFHR19] carry elements from both source groups $\mathbb{G}_1$ and $\mathbb{G}_2$ in the ciphertext (and this also seems like a typical property of encryption schemes in type-3 pairing settings). Hence, ciphertext re-randomization also needs to modify (and "refresh") elements from $\mathbb{G}_1$ *and* $\mathbb{G}_2$.

Now if we try to obtain tight security by applying the "adaptive partitioning" strategy to such a scheme, we would add consistency proofs to each ciphertext, essentially stating that all involved elements are chosen from the right joint distribution (*or* from some related distributions only used during the proof). Since the group elements in the scheme from [FFHR19] are tied together (even across $\mathbb{G}_1$ and $\mathbb{G}_2$), these proofs consider statements that involve both groups simultaneously.

This last property can be problematic for our overall proof strategy: following the adaptive partitioning strategy would mean to gradually inject entropy into the challenge ciphertexts, in our concrete case by introducing challenge-dependent randomness in the secret keys that are

used in the decryption oracle and to produce ("ill-formed") challenge ciphertexts. In case of the scheme from [FFHR19], this gradual randomization affects secret keys that refer to $\mathbb{G}_1$, as well as secret keys referring to $\mathbb{G}_2$.

The problem with this approach is now that randomizing both types of secret keys requires more entropy, and it is not clear where this entropy should come from. In fact, moving parts of the ciphertext outside a certain linear space (like we did with our value of $\vec{u}$ in our proof of Theorem 7.6.1) allows injecting the corresponding secret keys with additional entropy, but this entropy always refers to one single group ($\mathbb{G}_1$ or $\mathbb{G}_2$). Reconciling the entropy from both groups into one additional random function (like our function P_i) seems difficult.

In fact, an instantiation based on type-3 pairings seems to require new "entropy management" techniques to tightly randomize many challenge ciphertexts. We leave the construction of a tightly Rand-RCCA secure PKE scheme as an interesting open problem.

Appendices

Résumé en français

Le fil caché entre la cryptographie malléable et la cryptographie non malléable

Au cours de ces dernières décennies, notre société a drastiquement changé avec l'émergence du numérique. Presque chaque tâche, qu'il s'agisse d'envoyer des messages, d'effectuer des achats, de participer à des élections ou de signer des contrats, s'effectue désormais électroniquement. Cependant, cet immense terrain de jeu numérique offre également de nombreuses opportunités pour des activités malhonnêtes ou malveillantes. Il devient alors crucial de chercher à améliorer la sécurité de nos transactions. Le domaine de la *cryptographie* sert de base pour relever ce défi, en permettant l'exploration des principes et des limites de la cybersécurité.

Cryptographie : Un aperçu

La cryptographie a une longue et fascinante histoire qui remonte aux premiers siècles de l'humanité. Dans ce manuscrit, nous n'aborderons pas en détail le rôle que la cryptographie a joué tout au long de l'histoire, mais nous renvoyons le lecteur au célèbre livre *The Codebreakers* écrit par l'historien David Kahn [Kah67] pour un aperçu approfondi.

Cryptographie classique. Étymologiquement, le mot *cryptographie* vient du grec ancien κρυπτός ("caché, secret") et γράφειν ("écrire"). Jusqu'à la fin du 20e siècle, la cryptographie était effectivement synonyme de "chiffrement", à savoir le processus consistant à convertir un message (le "texte en clair") en un texte illisible (le "texte chiffré"), qui ne peut être lu qu'en inversant le processus (le "déchiffrement").

Les premiers systèmes cryptographiques étaient suffisamment simples pour être calculés et résolus à la main, et sont donc tombés en désuétude. Nous désignons désormais cette période sous le nom de *Cryptographie classique*, incluant les systèmes naïfs utilisés depuis l'époque grecque et romaine, les chiffres élaborés de la Renaissance, mais aussi les codes développés pendant la Seconde Guerre mondiale, comme la fameuse machine Enigma.

Cryptographie moderne. De nos jours, la cryptographie a considérablement évolué par rapport à ses racines historiques et englobe bien plus que la "criture secrète" : elle concerne des mécanismes permettant d'assurer l'intégrité des données ou des logiciels, de garantir la confidentialité en ligne, de gérer les identités numériques, de concevoir des protocoles sécurisés pour les élections électroniques, les systèmes de paiement, les applications de finance décentralisée (DeFi), et bien d'autres encore. Ainsi, la cryptographie moderne est la pratique et l'étude des techniques de communication sécurisée en présence de comportements adverses, et

se situe à l'intersection des disciplines des mathématiques, de l'informatique, de la sécurité de l'information, de la physique, et bien d'autres.

Dans les pages suivantes, nous proposons un aperçu très bref et de haut niveau de certaines des principales primitives et notions de sécurité en cryptographie, telles que les schémas de chiffrement, les preuves à divulgation nulle de connaissance et les protocoles sécurisés à plusieurs parties.

Schémas de chiffrement

Cryptographie à clé secrète. Le développement des systèmes cryptographiques modernes a commencé avec le travail de Feistel chez IBM au début des années 1970 [Fei73] et a culminé avec l'adoption du Data Encryption Standard (DES) en 1977, largement utilisé pour le chiffrement des données. Le DES utilise la cryptographie *à clé symétrique*, où la même clé est utilisée pour le chiffrement et le déchiffrement (d'où le nom symétrique). De toute évidence, dans ce cadre, la clé doit rester *secrète*, car elle est utilisée pour chiffrer et déchiffrer les données.

Cependant, un défi majeur réside dans la gestion des clés, chaque paire de parties communicantes devant idéalement disposer d'une clé unique, ce qui complique l'établissement sécurisé des clés sans un canal sécurisé préexistant.

Cryptographie à clé publique. Un moment clé de l'histoire de la cryptographie a eu lieu en 1976, lorsque Whitfield Diffie et Martin E. Hellman ont introduit la notion de cryptographie *à clé publique* dans leur article fondateur "New Directions in Cryptography" [DH76]. Cette innovation a résolu le problème de la gestion des clés et reposait sur la difficulté computationnelle du problème du logarithme discret (voir Section 2.3.1).

L'idée principale de la cryptographie à clé publique est de rompre la symétrie entre les clés de chiffrement et de déchiffrement. En particulier, il existe une paire de clés, l'une publique et l'autre privée. La clé publique permet à quiconque de chiffrer un message, qui ne pourra être déchiffré qu'avec la clé privée correspondante. La clé privée, quant à elle, est la seule à devoir être gardée secrète.

Bien que les premières implémentations pratiques du chiffrement à clé publique n'étaient pas disponibles à l'époque, l'idée a suscité un intérêt considérable et a stimulé le développement de la communauté cryptographique. En 1978, Rivest, Shamir et Adleman [RSA78] ont introduit le premier schéma de chiffrement à clé publique pratique. Leur schéma, désormais connu sous le nom de RSA, repose sur la difficulté de la factorisation de grands entiers, un problème qui a défié les mathématiciens pendant des siècles. Malgré le regain d'intérêt pour le développement de méthodes de factorisation plus efficaces, aucune avancée significative n'a compromis la sécurité du système cryptographique RSA, qui est encore largement utilisé en pratique aujourd'hui. Plus tard, de nouvelles constructions ont été proposées. Un exemple notable est le schéma de chiffrement ElGamal [ElG84], qui repose sur la difficulté du problème du logarithme discret.

Protocoles à plusieurs parties

Alors que les tâches cryptographiques traditionnelles se concentrent principalement sur l'assurance de la sécurité et de l'intégrité des communications entre deux utilisateurs, le modèle de sécurité représente souvent l'adversaire comme une entité opérant depuis *l'extérieur* du système des participants. Dans ce paradigme, en effet, les adversaires sont généralement perçus comme des

espions qui tentent d'intercepter et d'extraire des informations des échanges entre l'expéditeur et le récepteur. Cette perspective a servi de base pour les schémas de chiffrement et de nombreux protocoles cryptographiques conçus pour se protéger contre de telles menaces externes.

Cependant, à mesure que le domaine de la cryptographie continue d'évoluer, il devient de plus en plus important d'explorer des cadres alternatifs qui remettent en question cette vision traditionnelle. Un exemple notable remonte au travail fondamental de Yao [Yao86] qui introduisit la *calcul sécurisé à plusieurs parties* (MPC) [Yao86], un cadre cryptographique permettant à deux ou plusieurs parties de calculer conjointement une fonction sur leurs entrées respectives tout en préservant la confidentialité de ces entrées. Dans ce contexte, le modèle adversarial évolue considérablement ; plutôt que de simplement envisager des menaces externes, il faut prendre en compte la possibilité que certains participants puissent agir de manière malveillante ou déloyale.

Cela nécessite le développement de méthodes et de techniques capables de gérer différents types de comportements adverses, qu'ils proviennent d'un seul participant ou d'une coalition de participants qui peuvent *s'entendre* et être *corrompus* avant ou pendant le protocole. Cela implique également des efforts pour (re)définir le modèle adversarial et intégrer la dynamique de plusieurs parties, mais cela permet de développer des systèmes plus robustes, mieux équipés pour gérer les complexités des interactions réelles.

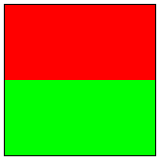
Nous nous référons à Section 2.2.3.5 pour un aperçu formel du modèle de *Composabilité Universelle* (UC) [Can01], un cadre notable qui offre de fortes garanties de sécurité pour les protocoles pouvant être utilisés dans le cadre de protocoles plus larges, et qui peuvent donc être *composés* en toute sécurité.

Systèmes de preuves et preuves à divulgation nulle de connaissance

Les preuves mathématiques traditionnelles sont soit évidentes d'elles-mêmes, soit basées sur des règles et axiomes établis ; de plus, elles peuvent être écrites et leur validité peut être vérifiée ligne par ligne.

Comme l'a expliqué Oded Goldreich [Gol95], la notion de preuve en cryptographie est sans doute différente : un système de preuve cryptographique est un *protocole interactif* où le prouveur cherche à convaincre le vérificateur de la vérité d'une affirmation. Dans le cas des preuves à divulgation nulle de connaissance, une exigence supplémentaire est que la preuve ne révèle rien au-delà de la validité de l'énoncé lui-même. Cela peut sembler paradoxal et contre-intuitif au début, c'est pourquoi nous proposons un exemple simple qui peut aider à comprendre le concept.[1]

Un exemple de preuve à divulgation nulle de connaissance. Imaginez un scénario où un prouveur, capable de différencier les couleurs, souhaite convaincre un vérificateur daltonien qu'une page particulière n'est pas monochromatique, par exemple, qu'elle contient deux couleurs comme ceci :



[1] Cet exemple est adapté d'une conférence de Shafi Goldwasser.

L'objectif du prouveur est de convaincre le vérificateur que cette page a deux couleurs sans transférer d'informations supplémentaires ou de "connaissances" (les couleurs elles-mêmes, dans ce cas, ou même la possibilité de les distinguer). Voici comment le processus se déroule :

1. Le vérificateur commence par lancer une pièce. Le vérificateur effectue cette action seul, sans révéler le résultat du lancer de la pièce au prouveur. Si la pièce tombe sur pile, il retournera la page ; si elle tombe sur face, il laissera la page inchangée.

2. Le prouveur examine la page. Puisqu'il sait à quoi la page ressemblait avant et qu'il peut distinguer les couleurs, il peut déterminer si le vérificateur l'a retournée ou non. Il vérifie si la page a été retournée et indique au vérificateur le résultat du lancer de la pièce.

3. Le vérificateur compare son propre lancer de la pièce avec celui deviné par le prouveur. S'ils correspondent, il conclut que le prouveur a pu savoir si la page a été retournée, ce qui implique qu'il doit y avoir deux couleurs sur la page. Ainsi, il accepte l'affirmation comme étant vraie. Sinon, il rejette l'affirmation.

Si la page a effectivement deux couleurs, le prouveur peut suivre ce procédé de manière fiable, et le vérificateur acceptera toujours l'affirmation : cette propriété est connue sous le nom de *complétude*.

Cependant, si la page a une seule couleur, le prouveur ne peut pas déterminer s'il l'a retournée ou non. Il pourrait deviner, ce qui donne une chance de 50% d'avoir raison. Ainsi, la probabilité qu'il convainque le vérificateur est au maximum de 50%, ce qui est dû à la chance, car l'affirmation du prouveur selon laquelle il y a deux couleurs est fausse : cette propriété est connue sous le nom de *robustesse*. La probabilité de cet événement défavorable peut être réduite en répétant le processus plusieurs fois, réduisant ainsi l'erreur de robustesse jusqu'à un facteur très faible.

Puisque le prouveur ne révèle rien d'autre que le résultat du lancer de la pièce, cette preuve est une preuve *à divulgation nulle de connaissance*.

Preuves de connaissance. Imaginez le scénario dans lequel un utilisateur souhaite s'authentifier sur un site web. L'utilisateur affirme que l'énoncé suivant est vrai : "il existe un mot de passe `pass` qui est le bon mot de passe pour le nom d'utilisateur `user`". En termes cryptographiques, `pass` est appelé le *témoin* associé à l'énoncé.

Étant donné que tous les utilisateurs enregistrés doivent avoir configuré un mot de passe, et que nous souhaitons nous assurer que seule une personne qui connaît réellement le mot de passe peut s'authentifier avec succès, prouver la véracité de cette affirmation ne suffit pas. C'est là qu'intervient la preuve de connaissance (*Proof of Knowledge* ou *PoK* en anglais).

Une PoK est une preuve qui permet au prouveur non seulement de convaincre le vérificateur que l'énoncé est vrai, mais aussi qu'il connaît un témoin valide associé à l'énoncé, ce qui, dans ce cas, correspond au mot de passe que l'utilisateur `user` a utilisé pour s'enregistrer sur le site.

Si la PoK est également à divulgation nulle de connaissance, cela signifie que bien que le vérificateur soit convaincu que l'utilisateur connaît bien le mot de passe, il ne reçoit aucune information supplémentaire sur le témoin lui-même. Cette solution est idéale dans ce cadre car elle garantit en plus que le mot de passe de l'utilisateur reste secret, même si un espion observe la communication entre l'utilisateur et le site web.

Preuves non interactives. Bien que dans les exemples ci-dessus, nous ayons présenté les preuves cryptographiques à divulgation nulle de connaissance comme des protocoles interactifs,

où les deux parties peuvent communiquer en temps réel, les rendant ainsi adaptées à des protocoles comme l'authentification par mot de passe, des preuves qui n'impliquent aucune interaction entre le prouveur et le vérificateur sont également possibles et connues sous le nom de *preuves non interactives à divulgation nulle de connaissance* (*Non-Interactive Zero-Knowledge Proofs* ou *NIZKs* en anglais). Dans les preuves non interactives, le prouveur génère une preuve qui peut être envoyée en un seul message au vérificateur. Cette caractéristique les rend attractives pour une grande variété d'applications, cependant, une preuve non interactive est intrinsèquement transférable [GOS06b], ce qui signifie qu'elle ne devrait pas être utilisée dans le cadre de l'authentification par mot de passe décrite ci-dessus : en gros, l'utilisateur `user`, connaissant le mot de passe correct, pourrait générer une PoK *statique* et non interactive et la partager avec d'autres utilisateurs, leur permettant ainsi de s'authentifier sur le site sans connaître eux-mêmes le mot de passe.

La transformation Fiat-Shamir. Dans certains protocoles interactifs, les messages du vérificateur sont véritablement aléatoires et indépendants des messages du prouveur : ces protocoles sont dits être des protocoles *à pièce publique*. Ces protocoles peuvent être convertis en protocoles non interactifs en utilisant la transformation Fiat-Shamir [FS87], une méthode heuristique qui remplace les messages aléatoires du vérificateur par une invocation d'un oracle aléatoire (*Random Oracle* ou *RO* en anglais) sur les messages du prouveur.

Un aperçu des modèles de sécurité et des configurations. Nous savons (voir [GO94]) qu'il n'est pas possible d'avoir des NIZKs pour des "tâches" non triviales sans faire d'hypothèses computationnelles (cf. Section 2.3), ce qui est appelé le Modèle Standard en cryptographie (cf. Section 2.2.3.1). Cependant, il est possible de construire des NIZKs pour une large gamme de tâches en assouplissant le modèle de sécurité ou en faisant des hypothèses supplémentaires.

Un cadre courant dans lequel les NIZKs peuvent être construits pour une large classe de tâches est lorsque le prouveur et le vérificateur ont tous deux accès à une chaîne de bits commune (*Common Reference String* our *CRS* en anglais) choisie par une partie de confiance : ce modèle est appelé le *modèle CRS*, introduit par Ivan Damgård [Dam00]. Si le CRS doit avoir une structure spécifique, on parle de *CRS structuré* (*Structured Reference String* en anglais), ou simplement de SRS. D'autre part, si le CRS est une chaîne de bits aléatoire, on parle parfois de *chaîne de référence uniforme*, ou modèle URS (*Uniform Reference String* ou *URS* en anglais). Pour certains schémas, il est nécessaire de générer un CRS pour chaque tâche différente, tandis que dans d'autres cas, un seul CRS peut être utilisé pour toutes les tâches : ce dernier modèle est appelé le *CRS universel*, dans le sens où toute tâche, jusqu'à un certain niveau de complexité, peut être gérée par un seul CRS. De plus, dans certains cas, il est possible de *mettre à jour* le CRS sans avoir à en générer un nouveau, ce qui est appelé le *modèle CRS actualisable* (*Updatable CRS* en anglais).

Preuves et arguments : une remarque sur la terminologie. Bien que dans cette introduction informelle, nous utilisions les termes de manière assez interchangeable, les preuves et les arguments sont différents. Un argument est une preuve dont la robustesse est supposée ne tenir que contre des proveurs malhonnêtes qui s'exécutent en temps polynomial, ce qui signifie qu'un proveur malhonnête ne peut pas convaincre le vérificateur d'une fausse affirmation *à moins* qu'il ne rompe une (presque toujours) hypothèse cryptographique difficile. Pour cette raison, les arguments sont parfois appelés *preuves computationnellement robustes*.

L'exemple simplifié du vérificateur daltonien mentionné ci-dessus est en réalité une preuve. Dans ce manuscrit, nous donnons une définition formelle des systèmes de preuves (cf. Defi-

nition 7.4.1). Cependant, la plupart du temps, nous nous concentrons sur les arguments non interactifs.

zkSNARKs

Un cas particulier d'intérêt est la classe des arguments de connaissance *succincts* et *non interactifs* à connaissance nulle (*Zero-Knowledge Non-Interactive Arguments of Knowledge* en anglais), ou simplement zkSNARKs. La raison pour laquelle les zkSNARKs sont si pertinents tant du point de vue pratique que théorique est qu'ils combinent les avantages des arguments de connaissance avec leur efficacité, c'est-à-dire les ressources nécessaires pour générer et vérifier la preuve, mais aussi leur *taille*. Un domaine de recherche actif sur les zkSNARKs a connu des progrès rapides dans plusieurs aspects, tels que l'efficacité [BCG+13, GGPR13, Gro10a, Gro16], la sécurité et la polyvalence de leurs configurations [BBHR19, GKM+18], et la composition des preuves [BDFG21, BCMS20]. Nous différons la définition formelle des zkSNARKs à Section 2.4.3.2.

Compilation des zkSNARKs. Une approche courante pour construire des zkSNARKs consiste d'abord à construire un protocole d'information théorique interactif à pièce publique, tel qu'une Preuve Oracle Interactive Polynomiale (*Polynomial Interactive Oracle Proof* ou *PIOP* en anglais), qui réalise la fonctionnalité souhaitée dans un modèle idéalisé, puis à supprimer la composante idéalisée en *la compilant* en un zkSNARK grâce à l'utilisation d'un primaire computationnellement sécurisé, tel qu'un engagement polynomiale, et en appliquant la transformation de Fiat-Shamir pour rendre la preuve non interactive. Nous développons davantage ce sujet dans Section 2.4.3.3.

zkVMs

Une approche populaire des SNARKs est celle des SNARKs pour *Machines Virtuelles* (ou SNARK VMs) qui, au cœur, consistent à prouver l'exécution d'un programme informatique, exprimé dans un jeu d'instructions prédéterminé, sur une abstraction de CPU. Cette conception présente plusieurs caractéristiques attrayantes : elle met à disposition tous les compilateurs d'optimisation existants pour des jeux d'instructions préexistants ; elle offre une excellente expérience pour les développeurs, rendant les SNARKs utilisables par toute personne capable d'écrire un programme informatique [AST24, Tha24b]. De nombreux SNARKs qui sont actuellement déployés en pratique suivent ce modèle de conception. Parmi les exemples, on trouve le Cairo-VM [GPR21], le projet RISC Zero [Zer], le Ceno de Scroll [LZZ+24], Polygon Miden [Lab] et bien d'autres. Parmi ces constructions, un exemple notable est Jolt [AST24], un SNARK pour VM basé sur l'approche *lookup-singularity* [Whi22], qui consiste à réduire l'exécution des opcodes dans une VM à une série de recherches dans des tables. Cette approche a un énorme potentiel d'adoption, étant à la fois simple, facile à étendre et à auditer. Elle permet également d'obtenir des proveurs extrêmement rapides (jusqu'à 2x plus rapides que l'état de l'art actuel [Tha24a]).

Questions de recherche

Dans ce manuscrit, nous explorons un certain nombre de questions de recherche qui sont pertinentes pour la malléabilité et la non-malléabilité des primitives et schémas cryptographiques

importants, et nous visons à fournir un récit cohérent qui intègre certains développements récents publiés simultanément ou après une partie du travail de ce manuscrit.

Nous analysons certaines des bases théoriques qui sous-tendent la malléabilité et ses implications pour la sécurité cryptographique : en cours de route, nous proposons soit de nouveaux schémas, soit nous revisitions l'analyse de sécurité de constructions existantes, et nous examinons les mécanismes qui peuvent transformer des schémas malléables en leurs homologues non malléables. En examinant les frontières et les connexions entre malléabilité et non-malléabilité, nous visons à fournir une compréhension plus profonde de la manière dont ces deux concepts peuvent coexister dans des protocoles cryptographiques robustes et efficaces.

Liste des publications. Les résultats présentés tout au long de ce manuscrit ont été publiés dans [FR22] (coécrit avec Antonio Faonio), [FHR23] (coécrit avec Antonio Faonio and Dennis Hofheinz), [FFK+23] (coécrit avec Antonio Faonio, Dario Fiore, Markus Kohlweiss and Michal Zajac) [FFR24] (coécrit avec Antonio Faonio and Dario Fiore), [CFR25] (coécrit avec Matteo Campanelli and Antonio Faonio).

Un résultat récent [BCC+24] (coécrit avec with Christian Badertscher, Matteo Campanelli, Michele Ciampi and Luisa Siniscalchi) sur la non-malléabilité des SNARKs n'a pas été inclus dans cette thèse.

Note à l'attention du lecteur. Dans les pages suivantes, nous développons les principales questions de recherche et les contributions qui sont présentées dans cette thèse. En particulier, lorsque nous examinerons l'état de l'art, la discussion deviendra plus technique. Ce changement est destiné aux lecteurs déjà familiers avec le domaine, car il repose sur des concepts et des méthodologies établis. Pour ceux qui pourraient avoir besoin de se familiariser avec certains aspects techniques, nous recommandons de consulter Chapter 2, qui fournit un aperçu fondamental des théories et terminologies pertinentes.

zkSNARKs simulables-extractibles

La plupart des zkSNARKs dans la littérature sont uniquement prouvés comme étant solidement basés sur la connaissance. Dans certains cas, cela est dû au fait que leurs preuves peuvent effectivement être malléables, par exemple dans [Gro16]. Dans d'autres cas, l'absence de preuve est due à la difficulté de la tâche, qui ne découle pas d'une simple extension de la preuve de solidité basée sur la connaissance, et nécessite donc une approche différente.

L'état des zkSNARKs simulables-extractibles. Jens Groth et Mary Maller proposent un zkSNARK simulable-extractible qui consiste en seulement 3 éléments de groupe [GM17], mais leur construction n'est ni universelle ni actualisable.

Le travail de Ganesh, Orlandi, Pancholi, Takahashi et Tschudi [GOP+22] montre que les Bulletproofs [BBB+18] sont non-malléables dans le Modèle de Groupe Algébrique (*Algebraic Group Model* ou *AGM* en anglais, voir Section 2.2.3.4). Quang Dao et Paul Grubbs montrent que Spartan [Set20] et Bulletproofs sont non-malléables même sans le AGM [DG23]. Ces deux travaux étendent le cadre introduit par Faust, Kohlweiss, Marson et Venturi dans [FKMV12] à la transformation de Fiat-Shamir appliquée aux arguments interactifs à plusieurs tours. Sur une voie similaire, le travail de Ganesh, Khoshakhlagh, Kohlweiss, Nitulescu et Zajac [GKK+22] montre la non-malléabilité pour Plonk [GWC19], Sonic [MBKM19] et Marlin [CHM+20]. Les

travaux [GKK+22, GOP+22] montrent que les arguments interactifs peuvent être simulables-extractibles après application de la transformation de Fiat-Shamir. En particulier, leur approche consiste à définir de nouvelles propriétés : *connaissance nulle sans porte dérobée* (*trapdoor-less zero-knowledge* en anglais), c'est-à-dire une connaissance nulle où le simulateur ne repose pas sur la porte dérobée du SRS mais sur la programmabilité de l'oracle aléatoire, et *réponse unique*, c'est-à-dire qu'à un moment donné du protocole, le prouveur devient un algorithme déterministe. Il est crucial que ces propriétés doivent être prouvées *au cas par cas* ; autrement dit, pour chaque candidat SNARK (même s'il résulte d'un compilateur générique), des efforts supplémentaires sont nécessaires pour montrer qu'il est simulable-extractible.

Un cadre général pour les zkSNARKs simulables-extractibles

Pouvons-nous trouver des conditions simples qui garantissent la non-malléabilité des zkSNARKs ?

Au départ, on peut se demander si la compilation naturelle des PIOPs vers les zkSNARKs conduit à des zkSNARKs simulables-extractibles si l'engagement polynomiale est simulable-extractible. Bien que cela soit possible, ce résultat ne couvrirait pas les zkSNARKs existants (ou instanciations), car l'engagement polynomiale KZG populaire [KZG10], qui conduit à certaines des instanciations les plus efficaces, est homomorphe. Pour y remédier, nous examinons plus en détail la malléabilité de l'engagement polynomiale KZG. Nous montrons ensuite que la compilation naturelle des PIOPs vers les zkSNARKs conduit à des zkSNARKs simulables-extractibles tant que l'engagement polynomiale possède une forme spéciale de simulabilité-extractibilité (ou de malléabilité contrôlée), ce que nous démontrerons pour l'engagement polynomiale KZG [KZG10], et que le PIOP satisfait à certaines conditions simples, respectées par des schémas populaires tels que Plonk [GWC19] et Marlin [CHM+20]. Nos résultats sur le compilateur ne reposent pas directement sur le AGM, ne nécessitent pas que le protocole soit sans porte dérobée et à connaissance nulle, ou ait la propriété de réponse unique, et utilisent une notion de robustesse plus standard, c'est-à-dire la robustesse de restauration de l'état des PIOPs.

Un autre cadre. Dans un travail concurrent, Kohlweiss, Pancholi et Takahashi [KPT23] ont également répondu à cette question et montré comment compiler des Preuves Holographiques Algébriques (*Algebraic Holographic Proofs* ou *AHP* en anglais) en zkSNARKs non malléables. Leur approche peut être considérée comme une extension des techniques de [GKO+23] et hérite de la nécessité d'une propriété de connaissance nulle sans porte dérobée. En même temps, les AHPs sont un cas particulier des PIOPs, et ainsi la classe de schémas que nous pouvons capturer est plus large.

zkSNARKs du monde réel

Notre résultat dans Chapter 3 fournit un cadre pour l'étude de la compilation naturelle des PIOPs vers les zkSNARKs. Cependant, les schémas qui offrent les meilleures performances et qui sont finalement implémentés dans les bibliothèques logicielles diffèrent de ceux obtenus par ce processus de compilation. En particulier, les versions *réelles* de schémas tels que Marlin ou Plonk utilisent une optimisation, le *truc de linéarisation* (également connu sous le nom d'optimisation de Mary Maller)[GWC19, OL], qui exploite les propriétés homomorphiques de l'engagement polynomiale KZG pour réduire le nombre d'éléments de champ dans la preuve.

Cette optimisation modifie toutefois l'algorithme de vérification des zkSNARKs d'une manière qui échappe à l'analyse de sécurité de Chapter 3 ; une limitation similaire s'applique au travail de Kohlweiss, Pancholi et Takahashi[KPT23], qui ne parvient pas à atteindre l'extractibilité simulable des versions "réelles" de Marlin et Plonk.

Que pouvons-nous dire sur la non-malléabilité des zkSNARKs optimisés ?

Dans Chapter 4, nous montrons comment résoudre les limitations ci-dessus et nous donnons la première preuve d'extractibilité simulable des versions "réelles" optimisées des zkSNARKs, qui incluent Plonk [GWC19], Marlin [CHM+20], Lunar [CFF+21] et Basilisk [RZ21].

En chemin, nous donnons la première analyse formelle du truc de linéarisation, en particulier de sa robustesse et de son extractibilité simulable dans le AGM (cf. Section 2.2.3.4).

SNARKs pour Machines Virtuelles

Aucun des résultats précédents sur l'extractibilité simulable des zkSNARKs ne couvre le cas des zkVMs. Une façon d'obtenir l'extractibilité simulable pour un zkVM est de le composer avec un autre zkSNARK, c'est-à-dire que nous pourrions utiliser un zkSNARK pour prouver la connaissance d'une preuve valide de zkVM (par exemple, le travail récent Testudo [CGG+23] compose Spartan avec Groth16 [Gro16], et certains schémas basés sur des repliements tels que Nova [KST22] suivent cette approche). Bien sûr, si ce zkSNARK est également simulable-extractible, alors il semble que nous obtenions le résultat maximal avec un effort minimal.

Bien que viable, cette approche de "ajout" de ZK par composition présente certains inconvénients théoriques et pratiques. En particulier, elle nécessiterait de représenter le vérificateur du zkVM dans un format comme R1CS ou Plonkish, ce qui pourrait être contraignant et limiter partiellement les avantages de l'auditabilité améliorée des zkVMs comme Jolt. De plus, cette procédure d'*arithmétisation* entraîne une instantiation directe de l'oracle aléatoire qui devient donc publique pour l'adversaire, ce qui pourrait mener à des schémas non sécurisés [CGH98].

Puisque les zkVMs sont au cœur de la conception de systèmes déployés avec des exigences de non-malléabilité

Que pouvons-nous dire sur la non-malléabilité des zkVMs ?

Dans Chapter 5, nous abordons ce problème en montrant qu'un zkVM modulaire inspiré de Jolt atteint l'extractibilité simulable sous des hypothèses minimales. Nous allons encore plus loin en explorant le cadre plus général de la composition de zkSNARKs (engagement-et-preuve).

PKE re-randomisable

Une classe intéressante de schémas de chiffrement à clé publique malléables est celle des schémas dits *re-randomisables*. Par re-randomisable, nous entendons que le texte chiffré c d'un message `msg` peut être transformé en un nouveau texte chiffré c', qui ressemble à un texte chiffré neuf, mais qui se déchiffre pour donner le *même* message. Cette fonctionnalité est utile pour implémenter des schémas de *réencryption par un proxy* et est présente dans des schémas PKE populaires, tels que le PKE ElGamal [ElG84].

La re-randomisation est en fait une forme simple de malléabilité. En général, nous souhaitons offrir une sécurité contre les attaques de malléabilité sur le message chiffré : un espion ne devrait pas être capable de *transformer* un texte chiffré d'un message `msg` en un texte chiffré

d'un autre message $\mathtt{msg}'$. Idéalement, nous aimerions disposer d'un schéma pouvant être re-randomisé, afin que nous puissions utiliser sa re-randomisation pour implémenter des protocoles comme *la réencryption par un proxy*, mais en même temps, nous souhaiterions prévenir toutes les autres attaques de malléabilité.

PKE Rand-RCCA. Jens Groth [Gro04] a introduit la notion de sécurité Re-randomisable Replayable CCA (Rand-RCCA) pour combler ce vide. Des travaux ultérieurs ont montré comment obtenir des schémas Rand-RCCA sous des hypothèses minimales [PR07], améliorer l'efficacité [FF20] ou renforcer les garanties de confidentialité [WCY+21].

Applications

Nous avons étudié les applications potentielles des schémas PKE Rand-RCCA. Le travail de Faonio *et al.* [FFHR19] a proposé un protocole de mix-net qui utilise les schémas PKE Rand-RCCA comme élément de base. Un Mix-net est un protocole qui permet à un ensemble d'expéditeurs d'envoyer des messages de manière anonyme, et trouve des applications dans différents domaines, notamment les e-mails anonymes, les paiements anonymes et le vote électronique, pour n'en citer que quelques-uns.

Pouvons-nous améliorer l'efficacité des protocoles qui reposent sur les schémas PKE Rand-RCCA ?

Dans Chapter 6, nous répondons affirmativement. Nous revisitons la conception du mix-net de [FFHR19] et proposons une instanciation plus efficace pour le protocole de mix-net basé sur leur schéma Rand-RCCA non publiquement vérifiable.

Sécurité concrète

George Orwell, dans son livre La Ferme des animaux, a écrit : "Tous les animaux sont égaux, mais certains sont plus égaux que d'autres."

D'une certaine manière, on pourrait aussi dire que tous les schémas PKE sont sécurisés, mais certains sont plus sécurisés que d'autres. En effet, bien qu'il soit souvent clair qu'un schéma PKE est sécurisé, il est souvent peu clair à quel point il offre une sécurité *concrète* (*tight* en anglais) lorsqu'il est utilisé dans le monde réel. Bellare, Boldyreva et Micali, dans leur travail fondamental [BBM00], ont étudié à quel point la sécurité d'un schéma de chiffrement se traduit par la confiance que nous avons vis-à-vis de l'hypothèse cryptographique sur laquelle il repose.

Plus précisément, une réduction de sécurité concrète garantit que pour toute attaque sur le schéma PKE, il existe une attaque sur l'hypothèse similaire à la fois en termes de complexité (c'est-à-dire le temps d'exécution, l'espace requis, etc.) et de probabilité de succès.

Ainsi, dans le cadre des réductions de sécurité concrètes, le nombre de textes chiffrés pris en compte par la définition de sécurité est important. À ce jour, de nombreux schémas PKE sécurisés contre les attaques CCA (cf. Section 1.2.1) ont été prouvés offrir une sécurité concrète dans les contextes multi-textes chiffrés et multi-utilisateurs : quelques exemples notables sont les travaux de [GHKW16, GHK17, HLLG19, Hof17, LJYP14, LPJY15]. Cependant, la sécurité concrète dans le contexte de la sécurité Rand-RCCA n'a pas été étudiée, bien que, en particulier, les cas d'utilisation Rand-RCCA ci-dessus comportent un grand nombre de textes chiffrés ou d'utilisateurs.

Existe-t-il un schéma PKE Rand-RCCA avec sécurité concrète ?

Dans Chapter 7, nous introduisons la notion de PKE Rand-RCCA multi-utilisateurs et multi-textes chiffrés et proposons la première construction d'un tel schéma PKE avec une réduction de sécurité concrète à une hypothèse computationnelle standard.

Bibliography

[ABC+22] Diego F. Aranha, Emil Madsen Bennedsen, Matteo Campanelli, Chaya Ganesh, Claudio Orlandi, and Akira Takahashi. ECLIPSE: Enhanced compiling method for Pedersen-committed zkSNARK engines. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022, Part I*, volume 13177 of *LNCS*, pages 584–614. Springer, Cham, March 2022. doi:10.1007/978-3-030-97121-2_21.

[Abe98] Masayuki Abe. Universally verifiable mix-net with verification work independent of the number of mix-servers. In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 437–447. Springer, Berlin, Heidelberg, May / June 1998. doi:10.1007/BFb0054144.

[Abe99] Masayuki Abe. Mix-networks on permutation networks. In Kwok-Yan Lam, Eiji Okamoto, and Chaoping Xing, editors, *ASIACRYPT'99*, volume 1716 of *LNCS*, pages 258–273. Springer, Berlin, Heidelberg, November 1999. doi:10.1007/978-3-540-48000-6_21.

[ABK+21] Michel Abdalla, Manuel Barbosa, Jonathan Katz, Julian Loss, and Jiayu Xu. Algebraic adversaries in the universal composability framework. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part III*, volume 13092 of *LNCS*, pages 311–341. Springer, Cham, December 2021. doi:10.1007/978-3-030-92078-4_11.

[ABP15] Michel Abdalla, Fabrice Benhamouda, and David Pointcheval. Disjunctions for hash proof systems: New constructions and applications. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 69–100. Springer, Berlin, Heidelberg, April 2015. doi:10.1007/978-3-662-46803-6_3.

[ACK21] Thomas Attema, Ronald Cramer, and Lisa Kohl. A compressed Σ-protocol theory for lattices. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 549–579, Virtual Event, August 2021. Springer, Cham. doi:10.1007/978-3-030-84245-1_19.

[AH01] Masayuki Abe and Fumitaka Hoshino. Remarks on mix-network based on permutation networks. In Kwangjo Kim, editor, *PKC 2001*, volume 1992 of *LNCS*, pages 317–324. Springer, Berlin, Heidelberg, February 2001. doi:10.1007/3-540-44586-2_23.

[AJOR18] Masayuki Abe, Charanjit S. Jutla, Miyako Ohkubo, and Arnab Roy. Improved (almost) tightly-secure simulation-sound QA-NIZK with applications. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part I*, volume 11272 of *LNCS*, pages 627–656. Springer, Cham, December 2018. doi:10.1007/978-3-030-03326-2_21.

[ark21] arkworks. Diagram of marlin's prover and verifier, including optimizations, 2021. https://github.com/arkworks-rs/marlin/blob/master/diagram/diagram.pdf.

[ARS20] Behzad Abdolmaleki, Sebastian Ramacher, and Daniel Slamanig. Lift-and-shift: Obtaining simulation extractable subversion and updatable SNARKs generically. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1987–2005. ACM Press, November 2020. doi:10.1145/3372297.3417228.

[AST24] Arasu Arun, Srinath T. V. Setty, and Justin Thaler. Jolt: SNARKs for virtual machines via lookups. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part VI*, volume 14656 of *LNCS*, pages 3–33. Springer, Cham, May 2024. doi:10.1007/978-3-031-58751-1_1.

[BB04] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer, Berlin, Heidelberg, May 2004. doi:10.1007/978-3-540-24676-3_4.

[BB08] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, April 2008. doi:10.1007/s00145-007-9005-7.

[BBB+18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018. doi:10.1109/SP.2018.00020.

[BBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046, 2018. URL: https://eprint.iacr.org/2018/046.

[BBHR19] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable zero knowledge with no trusted setup. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 701–732. Springer, Cham, August 2019. doi:10.1007/978-3-030-26954-8_23.

[BBM00] Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 259–274. Springer, Berlin, Heidelberg, May 2000. doi:10.1007/3-540-45539-6_18.

[BBS98] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 127–144. Springer, Berlin, Heidelberg, May / June 1998. doi:10.1007/BFb0054122.

[BCC+24] Christian Badertscher, Matteo Campanelli, Michele Ciampi, Luigi Russo, and Luisa Siniscalchi. Universally composable SNARKs with transparent setup without programmable random oracle. Cryptology ePrint Archive, Report 2024/1549, 2024. URL: https://eprint.iacr.org/2024/1549.

[BCCT12] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In Shafi Goldwasser, editor, *ITCS 2012*, pages 326–349. ACM, January 2012. doi:10.1145/2090236.2090263.

[BCF+21] Daniel Benarroch, Matteo Campanelli, Dario Fiore, Kobi Gurkan, and Dimitris Kolonelos. Zero-knowledge proofs for set membership: Efficient, succinct, modular. In Nikita Borisov and Claudia Díaz, editors, *FC 2021, Part I*, volume 12674 of *LNCS*, pages 393–414. Springer, Berlin, Heidelberg, March 2021. doi:10.1007/978-3-662-64322-8_19.

[BCG+13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. SNARKs for C: Verifying program executions succinctly and in zero knowledge. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 90–108. Springer, Berlin, Heidelberg, August 2013. doi:10.1007/978-3-642-40084-1_6.

[BCG+18] Jonathan Bootle, Andrea Cerulli, Jens Groth, Sune K. Jakobsen, and Mary Maller. Arya: Nearly linear-time zero-knowledge proofs for correct program execution. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part I*, volume 11272 of *LNCS*, pages 595–626. Springer, Cham, December 2018. doi:10.1007/978-3-030-03326-2_20.

[BCKL08] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. P-signatures and noninteractive anonymous credentials. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 356–374. Springer, Berlin, Heidelberg, March 2008. doi:10.1007/978-3-540-78524-8_20.

[BCMS20] Benedikt Bünz, Alessandro Chiesa, Pratyush Mishra, and Nicholas Spooner. Recursive proof composition from accumulation schemes. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 1–18. Springer, Cham, November 2020. doi:10.1007/978-3-030-64378-2_1.

[BCR+19] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 103–128. Springer, Cham, May 2019. doi:10.1007/978-3-030-17653-2_4.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 31–60. Springer, Berlin, Heidelberg, October / November 2016. doi:10.1007/978-3-662-53644-5_2.

[BDFG20] Dan Boneh, Justin Drake, Ben Fisch, and Ariel Gabizon. Efficient polynomial commitment schemes for multiple points and polynomials. Cryptology ePrint Archive, Report 2020/081, 2020. URL: https://eprint.iacr.org/2020/081.

[BDFG21] Dan Boneh, Justin Drake, Ben Fisch, and Ariel Gabizon. Halo infinite: Proof-carrying data from additive polynomial commitments. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 649–680, Virtual Event, August 2021. Springer, Cham. doi:10.1007/978-3-030-84242-0_23.

[BDP00] Joan Boyar, Ivan Damgård, and René Peralta. Short non-interactive cryptographic proofs. *Journal of Cryptology*, 13(4):449–472, September 2000. doi:10.1007/s001450010011.

[BDS98] Mike Burmester, Yvo Desmedt, and Jennifer Seberry. Equitable key escrow with limited time span (or, how to enforce time expiration cryptographically). In Kazuo Ohta and Dingyi Pei, editors, *ASIACRYPT'98*, volume 1514 of *LNCS*, pages 380–391. Springer, Berlin, Heidelberg, October 1998. doi:10.1007/3-540-49649-1_30.

[BEG+91] Manuel Blum, William S. Evans, Peter Gemmell, Sampath Kannan, and Moni Naor. Checking the correctness of memories. In *32nd FOCS*, pages 90–99. IEEE Computer Society Press, October 1991. doi:10.1109/SFCS.1991.185352.

[Ben06] Josh Benaloh. Simple verifiable elections. In *2006 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT 06)*, Vancouver, B.C., August 2006. USENIX Association. URL: https://www.usenix.org/conference/evt-06/simple-verifiable-elections.

[BFHK23] Balthazar Bauer, Pooya Farshim, Patrick Harasser, and Markulf Kohlweiss. The uber-knowledge assumption: A bridge to the AGM. Cryptology ePrint Archive, Report 2023/1601, 2023. URL: https://eprint.iacr.org/2023/1601.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th ACM STOC*, pages 103–112. ACM Press, May 1988. doi:10.1145/62212.62222.

[BFS20] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent SNARKs from DARK compilers. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 677–706. Springer, Cham, May 2020. doi:10.1007/978-3-030-45721-1_24.

[BG12] Stephanie Bayer and Jens Groth. Efficient zero-knowledge argument for correctness of a shuffle. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 263–280. Springer, Berlin, Heidelberg, April 2012. doi:10.1007/978-3-642-29011-4_17.

[BGKS20] Eli Ben-Sasson, Lior Goldberg, Swastik Kopparty, and Shubhangi Saraf. DEEP-FRI: Sampling outside the box improves soundness. In Thomas Vidick, editor, *ITCS 2020*, volume 151, pages 5:1–5:32. LIPIcs, January 2020. doi:10.4230/LIPIcs.ITCS.2020.5.

[BHK15] Mihir Bellare, Dennis Hofheinz, and Eike Kiltz. Subtleties in the definition of IND-CCA: When and how should challenge decryption be disallowed? *Journal of Cryptology*, 28(1):29–48, January 2015. doi:10.1007/s00145-013-9167-4.

[Boy08] Xavier Boyen. The uber-assumption family (invited talk). In Steven D. Galbraith and Kenneth G. Paterson, editors, *PAIRING 2008*, volume 5209 of *LNCS*, pages 39–56. Springer, Berlin, Heidelberg, September 2008. doi:10.1007/978-3-540-85538-5_3.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993. doi:10.1145/168588.168596.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001. doi:10.1109/SFCS.2001.959888.

[CBBZ23] Binyi Chen, Benedikt Bünz, Dan Boneh, and Zhenfei Zhang. HyperPlonk: Plonk with linear-time prover and high-degree custom gates. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part II*, volume 14005 of *LNCS*, pages 499–530. Springer, Cham, April 2023. doi:10.1007/978-3-031-30617-4_17.

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 19–40. Springer, Berlin, Heidelberg, August 2001. doi:10.1007/3-540-44647-8_2.

[CF24] Alessandro Chiesa and Giacomo Fenzi. zkSNARKs in the ROM with unconditional UC-security. In Elette Boyle and Mohammad Mahmoody, editors, *TCC 2024, Part I*, volume 15364 of *LNCS*, pages 67–89. Springer, Cham, December 2024. doi:10.1007/978-3-031-78011-0_3.

[CFF+21] Matteo Campanelli, Antonio Faonio, Dario Fiore, Anaïs Querol, and Hadrián Rodríguez. Lunar: A toolbox for more efficient universal and updatable zkSNARKs and commit-and-prove extensions. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part III*, volume 13092 of *LNCS*, pages 3–33. Springer, Cham, December 2021. doi:10.1007/978-3-030-92078-4_1.

[CFH+22] Matteo Campanelli, Dario Fiore, Semin Han, Jihye Kim, Dimitris Kolonelos, and Hyunok Oh. Succinct zero-knowledge batch proofs for set accumulators. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 455–469. ACM Press, November 2022. doi:10.1145/3548606.3560677.

[CFQ19] Matteo Campanelli, Dario Fiore, and Anaïs Querol. LegoSNARK: Modular design and composition of succinct zero-knowledge proofs. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2075–2092. ACM Press, November 2019. doi:10.1145/3319535.3339820.

[CFR25] Matteo Campanelli, Antonio Faonio, and Luigi Russo. SNARKs for virtual machines are non-malleable. In Serge Fehr and Pierre-Alain Fouque, editors, *EUROCRYPT 2025, Part IV*, volume 15604 of *LNCS*, pages 153–183. Springer, Cham, May 2025. doi:10.1007/978-3-031-91134-7_6.

[CGG+23] Matteo Campanelli, Nicolas Gailly, Rosario Gennaro, Philipp Jovanovic, Mara Mihali, and Justin Thaler. Testudo: Linear time prover SNARKs with constant size proofs and square root size universal setup. In Abdelrahaman Aly and Mehdi Tibouchi, editors, *LATINCRYPT 2023*, volume 14168 of *LNCS*, pages 331–351. Springer, Cham, October 2023. doi:10.1007/978-3-031-44469-2_17.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th ACM STOC*, pages 209–218. ACM Press, May 1998. doi:10.1145/276698.276741.

[Cha81] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, February 1981. URL: http://doi.acm.org/10.1145/358549.358563, doi:10.1145/358549.358563.

[CHM+20] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Psi Vesely, and Nicholas P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 738–768. Springer, Cham, May 2020. doi:10.1007/978-3-030-45721-1_26.

[CKLM12] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Malleable proof systems and applications. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 281–300. Springer, Berlin, Heidelberg, April 2012. doi:10.1007/978-3-642-29011-4_18.

[CKN03] Ran Canetti, Hugo Krawczyk, and Jesper Buus Nielsen. Relaxing chosen-ciphertext security. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 565–582. Springer, Berlin, Heidelberg, August 2003. doi:10.1007/978-3-540-45146-4_33.

[CL06] Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 78–96. Springer, Berlin, Heidelberg, August 2006. doi:10.1007/11818175_5.

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 13–25. Springer, Berlin, Heidelberg, August 1998. doi:10.1007/BFb0055717.

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer, Berlin, Heidelberg, April / May 2002. doi:10.1007/3-540-46035-7_4.

[Dam93] Ivan Damgård. Non-interactive circuit based proofs and non-interactive perfect zero-knowledge with proprocessing. In Rainer A. Rueppel, editor, *EUROCRYPT'92*, volume 658 of *LNCS*, pages 341–355. Springer, Berlin, Heidelberg, May 1993. doi:10.1007/3-540-47555-9_28.

[Dam00] Ivan Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 418–430. Springer, Berlin, Heidelberg, May 2000. doi:10.1007/3-540-45539-6_30.

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *23rd ACM STOC*, pages 542–552. ACM Press, May 1991. doi:10.1145/103418.103474.

[DDO+01] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 566–598. Springer, Berlin, Heidelberg, August 2001. doi:10.1007/3-540-44647-8_33.

[Den02] Alexander W. Dent. Adapting the weaknesses of the random oracle model to the generic group model. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 100–109. Springer, Berlin, Heidelberg, December 2002. doi:10.1007/3-540-36178-2_6.

[DFH12] Ivan Damgård, Sebastian Faust, and Carmit Hazay. Secure two-party computation with low communication. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 54–74. Springer, Berlin, Heidelberg, March 2012. doi:10.1007/978-3-642-28914-9_4.

[DG23] Quang Dao and Paul Grubbs. Spartan and bulletproofs are simulation-extractable (for free!). In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part II*, volume 14005 of *LNCS*, pages 531–562. Springer, Cham, April 2023. doi:10.1007/978-3-031-30617-4_18.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. doi:10.1109/TIT.1976.1055638.

[DHLW10] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Efficient public-key cryptography in the presence of key leakage. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 613–631. Springer, Berlin, Heidelberg, December 2010. doi:10.1007/978-3-642-17373-8_35.

[DMP90] Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Non-interactive zero-knowledge with preprocessing. In Shafi Goldwasser, editor, *CRYPTO'88*, volume

403 of *LNCS*, pages 269–282. Springer, New York, August 1990. doi:10.1007/0-387-34799-2_21.

[DMS16] Yevgeniy Dodis, Ilya Mironov, and Noah Stephens-Davidowitz. Message transmission with reverse firewalls—secure communication on corrupted machines. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 341–372. Springer, Berlin, Heidelberg, August 2016. doi:10.1007/978-3-662-53018-4_13.

[DOTV22] Cyprien Delpech de Saint Guilhem, Emmanuela Orsini, Titouan Tanguy, and Michiel Verbauwhede. Efficient proof of RAM programs from any public-coin zero-knowledge system. Cryptology ePrint Archive, Report 2022/313, 2022. URL: https://eprint.iacr.org/2022/313.

[EG14] Alex Escala and Jens Groth. Fine-tuning Groth-Sahai proofs. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 630–649. Springer, Berlin, Heidelberg, March 2014. doi:10.1007/978-3-642-54631-0_36.

[EHK+13] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Berlin, Heidelberg, August 2013. doi:10.1007/978-3-642-40084-1_8.

[ElG84] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 10–18. Springer, Berlin, Heidelberg, August 1984. doi:10.1007/3-540-39568-7_2.

[Fei73] Horst Feistel. Cryptography and computer privacy. *Scientific American*, 228(5):15–23, 1973.

[FF20] Antonio Faonio and Dario Fiore. Improving the efficiency of re-randomizable and replayable CCA secure public key encryption. In Mauro Conti, Jianying Zhou, Emiliano Casalicchio, and Angelo Spognardi, editors, *ACNS 2020, Part I*, volume 12146 of *LNCS*, pages 271–291. Springer, Cham, October 2020. doi:10.1007/978-3-030-57808-4_14.

[FFHR19] Antonio Faonio, Dario Fiore, Javier Herranz, and Carla Ràfols. Structure-preserving and re-randomizable RCCA-secure public key encryption and its applications. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 159–190. Springer, Cham, December 2019. doi:10.1007/978-3-030-34618-8_6.

[FFK+23] Antonio Faonio, Dario Fiore, Markulf Kohlweiss, Luigi Russo, and Michal Zajac. From polynomial IOP and commitments to non-malleable zkSNARKs. In Guy N. Rothblum and Hoeteck Wee, editors, *TCC 2023, Part III*, volume 14371 of *LNCS*, pages 455–485. Springer, Cham, November / December 2023. doi:10.1007/978-3-031-48621-0_16.

[FFR24] Antonio Faonio, Dario Fiore, and Luigi Russo. Real-world universal zkSNARKs are non-malleable. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024*, pages 3138–3151. ACM Press, October 2024. doi:10.1145/3658644.3690351.

[FHR23] Antonio Faonio, Dennis Hofheinz, and Luigi Russo. Almost tightly-secure re-randomizable and replayable CCA-secure public key encryption. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023, Part II*, volume 13941 of *LNCS*, pages 275–305. Springer, Cham, May 2023. doi:10.1007/978-3-031-31371-4_10.

[Fis05] Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 152–168. Springer, Berlin, Heidelberg, August 2005. doi:10.1007/11535218_10.

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Cham, August 2018. doi:10.1007/978-3-319-96881-0_2.

[FKMV12] Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the non-malleability of the Fiat-Shamir transform. In Steven D. Galbraith and Mridul Nandi, editors, *INDOCRYPT 2012*, volume 7668 of *LNCS*, pages 60–79. Springer, Berlin, Heidelberg, December 2012. doi:10.1007/978-3-642-34931-7_5.

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *31st FOCS*, pages 308–317. IEEE Computer Society Press, October 1990. doi:10.1109/FSCS.1990.89549.

[FLSZ17] Prastudy Fauzi, Helger Lipmaa, Janno Siim, and Michal Zajac. An efficient pairing-based shuffle argument. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part II*, volume 10625 of *LNCS*, pages 97–127. Springer, Cham, December 2017. doi:10.1007/978-3-319-70697-9_4.

[FR22] Antonio Faonio and Luigi Russo. Mix-nets from re-randomizable and replayable CCA-secure public-key encryption. In Clemente Galdi and Stanislaw Jarecki, editors, *SCN 22*, volume 13409 of *LNCS*, pages 172–196. Springer, Cham, September 2022. doi:10.1007/978-3-031-14791-3_8.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Berlin, Heidelberg, August 1987. doi:10.1007/3-540-47721-7_12.

[FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *22nd ACM STOC*, pages 416–426. ACM Press, May 1990. doi:10.1145/100216.100272.

[FS01] Jun Furukawa and Kazue Sako. An efficient scheme for proving a shuffle. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 368–387. Springer, Berlin, Heidelberg, August 2001. doi:10.1007/3-540-44647-8_22.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645. Springer, Berlin, Heidelberg, May 2013. doi:10.1007/978-3-642-38348-9_37.

[GHK17] Romain Gay, Dennis Hofheinz, and Lisa Kohl. Kurosawa-desmedt meets tight security. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 133–160. Springer, Cham, August 2017. doi:10.1007/978-3-319-63697-9_5.

[GHKP18] Romain Gay, Dennis Hofheinz, Lisa Kohl, and Jiaxin Pan. More efficient (almost) tightly secure structure-preserving signatures. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 230–258. Springer, Cham, April / May 2018. doi:10.1007/978-3-319-78375-8_8.

[GHKW16] Romain Gay, Dennis Hofheinz, Eike Kiltz, and Hoeteck Wee. Tightly CCA-secure encryption without pairings. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 1–27. Springer, Berlin, Heidelberg, May 2016. doi:10.1007/978-3-662-49890-3_1.

[GI08] Jens Groth and Yuval Ishai. Sub-linear zero-knowledge argument for correctness of a shuffle. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 379–396. Springer, Berlin, Heidelberg, April 2008. doi:10.1007/978-3-540-78967-3_22.

[GJS11] Sanjam Garg, Abhishek Jain, and Amit Sahai. Leakage-resilient zero knowledge. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 297–315. Springer, Berlin, Heidelberg, August 2011. doi:10.1007/978-3-642-22792-9_17.

[GKK+22] Chaya Ganesh, Hamidreza Khoshakhlagh, Markulf Kohlweiss, Anca Nitulescu, and Michal Zajac. What makes fiat-shamir zkSNARKs (updatable SRS) simulation extractable? In Clemente Galdi and Stanislaw Jarecki, editors, *SCN 22*, volume 13409 of *LNCS*, pages 735–760. Springer, Cham, September 2022. doi:10.1007/978-3-031-14791-3_32.

[GKM+18] Jens Groth, Markulf Kohlweiss, Mary Maller, Sarah Meiklejohn, and Ian Miers. Updatable and universal common reference strings with applications to zk-SNARKs. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 698–728. Springer, Cham, August 2018. doi:10.1007/978-3-319-96878-0_24.

[GKO+23] Chaya Ganesh, Yashvanth Kondi, Claudio Orlandi, Mahak Pancholi, Akira Takahashi, and Daniel Tschudi. Witness-succinct universally-composable SNARKs.

In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part II*, volume 14005 of *LNCS*, pages 315–346. Springer, Cham, April 2023. doi:10.1007/978-3-031-30617-4_11.

[GM17] Jens Groth and Mary Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 581–612. Springer, Cham, August 2017. doi:10.1007/978-3-319-63715-0_20.

[GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, December 1994. doi:10.1007/BF00195207.

[Gol95] Oded Goldreich. Foundations of cryptography (fragments of a book). Electronic Colloquium on Computational Complexity, 1995.

[GOP+22] Chaya Ganesh, Claudio Orlandi, Mahak Pancholi, Akira Takahashi, and Daniel Tschudi. Fiat-shamir bulletproofs are non-malleable (in the algebraic group model). In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 397–426. Springer, Cham, May / June 2022. doi:10.1007/978-3-031-07085-3_14.

[GOS06a] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for NIZK. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 97–111. Springer, Berlin, Heidelberg, August 2006. doi:10.1007/11818175_6.

[GOS06b] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 339–358. Springer, Berlin, Heidelberg, May / June 2006. doi:10.1007/11761679_21.

[GPR21] Lior Goldberg, Shahar Papini, and Michael Riabzev. Cairo – a Turing-complete STARK-friendly CPU architecture. Cryptology ePrint Archive, Report 2021/1063, 2021. URL: https://eprint.iacr.org/2021/1063.

[Gro03] Jens Groth. A verifiable secret shuffle of homomorphic encryptions. In Yvo Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 145–160. Springer, Berlin, Heidelberg, January 2003. doi:10.1007/3-540-36288-6_11.

[Gro04] Jens Groth. Rerandomizable and replayable adaptive chosen ciphertext attack secure cryptosystems. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 152–170. Springer, Berlin, Heidelberg, February 2004. doi:10.1007/978-3-540-24638-1_9.

[Gro10a] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340. Springer, Berlin, Heidelberg, December 2010. doi:10.1007/978-3-642-17373-8_19.

[Gro10b] Jens Groth. A verifiable secret shuffle of homomorphic encryptions. *Journal of Cryptology*, 23(4):546–579, October 2010. doi:10.1007/s00145-010-9067-9.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Berlin, Heidelberg, May 2016. doi:10.1007/978-3-662-49896-5_11.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Berlin, Heidelberg, April 2008. doi:10.1007/978-3-540-78967-3_24.

[GT21] Ashrujit Ghoshal and Stefano Tessaro. Tight state-restoration soundness in the algebraic group model. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part III*, volume 12827 of *LNCS*, pages 64–93, Virtual Event, August 2021. Springer, Cham. doi:10.1007/978-3-030-84252-9_3.

[GWC19] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: Permutations over Lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019. URL: https://eprint.iacr.org/2019/953.

[HHH+14] Gottfried Herold, Julia Hesse, Dennis Hofheinz, Carla Ràfols, and Andy Rupp. Polynomial spaces: A new framework for composite-to-prime-order transformations. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 261–279. Springer, Berlin, Heidelberg, August 2014. doi:10.1007/978-3-662-44371-2_15.

[HLLG19] Shuai Han, Shengli Liu, Lin Lyu, and Dawu Gu. Tight leakage-resilient CCA-security from quasi-adaptive hash proof system. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 417–447. Springer, Cham, August 2019. doi:10.1007/978-3-030-26951-7_15.

[Hof17] Dennis Hofheinz. Adaptive partitioning. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 489–518. Springer, Cham, April / May 2017. doi:10.1007/978-3-319-56617-7_17.

[Ish19] Yuval Ishai. Efficient zero-knowledge proofs: A modular approach. https://simons.berkeley.edu/talks/tbd-79. Also see https://zkproof.org/2020/08/12/information-theoretic-proof-systems/, 2019.

[Ish20] Yuval Ishai. Zero-Knowledge Proofs from Information-Theoretic Proof Systems - Part I. ZKProof.org, Blog entry, August 2020. Also see https://zkproof.org/2020/08/12/information-theoretic-proof-systems/.

[JM99] Markus Jakobsson and David M'Raïhi. Mix-based electronic payments. In Stafford E. Tavares and Henk Meijer, editors, *SAC 1998*, volume 1556 of *LNCS*, pages 157–173. Springer, Berlin, Heidelberg, August 1999. doi:10.1007/3-540-48892-8_13.

[JR13] Charanjit S. Jutla and Arnab Roy. Shorter quasi-adaptive NIZK proofs for linear subspaces. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 1–20. Springer, Berlin, Heidelberg, December 2013. doi:10.1007/978-3-642-42033-7_1.

[JR14] Charanjit S. Jutla and Arnab Roy. Switching lemma for bilinear tests and constant-size NIZK proofs for linear subspaces. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 295–312. Springer, Berlin, Heidelberg, August 2014. doi:10.1007/978-3-662-44381-1_17.

[Kah67] David Kahn. *The Codebreakers*. Macmillan, 1967.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *24th ACM STOC*, pages 723–732. ACM Press, May 1992. doi:10.1145/129712.129782.

[KM15] Neal Koblitz and Alfred J. Menezes. The random oracle model: a twenty-year retrospective. *DCC*, 77(2-3):587–610, 2015. doi:10.1007/s10623-015-0094-2.

[KP98] Joe Kilian and Erez Petrank. An efficient noninteractive zero-knowledge proof system for NP with general assumptions. *Journal of Cryptology*, 11(1):1–27, January 1998. doi:10.1007/s001459900032.

[KPT23] Markulf Kohlweiss, Mahak Pancholi, and Akira Takahashi. How to compile polynomial IOP into simulation-extractable SNARKs: A modular approach. In Guy N. Rothblum and Hoeteck Wee, editors, *TCC 2023, Part III*, volume 14371 of *LNCS*, pages 486–512. Springer, Cham, November / December 2023. doi:10.1007/978-3-031-48621-0_17.

[KST22] Abhiram Kothapalli, Srinath Setty, and Ioanna Tzialla. Nova: Recursive zero-knowledge arguments from folding schemes. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part IV*, volume 13510 of *LNCS*, pages 359–388. Springer, Cham, August 2022. doi:10.1007/978-3-031-15985-5_13.

[KW15] Eike Kiltz and Hoeteck Wee. Quasi-adaptive NIZK for linear subspaces revisited. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 101–128. Springer, Berlin, Heidelberg, April 2015. doi:10.1007/978-3-662-46803-6_4.

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194. Springer, Berlin, Heidelberg, December 2010. doi:10.1007/978-3-642-17373-8_11.

[Lab] Polygon Labs. Polygon Miden. https://github.com/0xPolygonMiden.

[Lee21] Jonathan Lee. Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part II*, volume 13043 of *LNCS*, pages 1–34. Springer, Cham, November 2021. doi:10.1007/978-3-030-90453-1_1.

[Lip12] Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 169–189. Springer, Berlin, Heidelberg, March 2012. doi:10.1007/978-3-642-28914-9_10.

[LJYP14] Benoît Libert, Marc Joye, Moti Yung, and Thomas Peters. Concise multi-challenge CCA-secure encryption and signatures with almost tight security. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 1–21. Springer, Berlin, Heidelberg, December 2014. doi:10.1007/978-3-662-45608-8_1.

[LPJY15] Benoît Libert, Thomas Peters, Marc Joye, and Moti Yung. Compactly hiding linear spans - tightly secure constant-size simulation-sound QA-NIZK proofs and applications. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 681–707. Springer, Berlin, Heidelberg, November / December 2015. doi:10.1007/978-3-662-48797-6_28.

[LPQ17] Benoît Libert, Thomas Peters, and Chen Qian. Structure-preserving chosen-ciphertext security with shorter verifiable ciphertexts. In Serge Fehr, editor, *PKC 2017, Part I*, volume 10174 of *LNCS*, pages 247–276. Springer, Berlin, Heidelberg, March 2017. doi:10.1007/978-3-662-54365-8_11.

[LPS23] Helger Lipmaa, Roberto Parisella, and Janno Siim. Algebraic group model with oblivious sampling. In Guy N. Rothblum and Hoeteck Wee, editors, *TCC 2023, Part IV*, volume 14372 of *LNCS*, pages 363–392. Springer, Cham, November / December 2023. doi:10.1007/978-3-031-48624-1_14.

[LZZ+24] Tianyi Liu, Zhenfei Zhang, Yuncong Zhang, Wenqing Hu, and Ye Zhang. Ceno: Non-uniform, segment and parallel zero-knowledge virtual machine. Cryptology ePrint Archive, Paper 2024/387, 2024. URL: https://eprint.iacr.org/2024/387.

[Mau05] Ueli M. Maurer. Abstract models of computation in cryptography (invited paper). In Nigel P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *LNCS*, pages 1–12. Springer, Berlin, Heidelberg, December 2005. doi:10.1007/11586821_1.

[MBKM19] Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2111–2128. ACM Press, November 2019. doi:10.1145/3319535.3339817.

[Mic94] Silvio Micali. CS proofs (extended abstracts). In *35th FOCS*, pages 436–453. IEEE Computer Society Press, November 1994. doi:10.1109/SFCS.1994.365746.

[MRV16] Paz Morillo, Carla Ràfols, and Jorge Luis Villar. The kernel matrix Diffie-Hellman assumption. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016,*

Part I, volume 10031 of *LNCS*, pages 729–758. Springer, Berlin, Heidelberg, December 2016. doi:10.1007/978-3-662-53887-6_27.

[MW96] Ueli M. Maurer and Stefan Wolf. Diffie-Hellman oracles. In Neal Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 268–282. Springer, Berlin, Heidelberg, August 1996. doi:10.1007/3-540-68697-5_21.

[NAP+14] Muhammad Naveed, Shashank Agrawal, Manoj Prabhakaran, XiaoFeng Wang, Erman Ayday, Jean-Pierre Hubaux, and Carl A. Gunter. Controlled functional encryption. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 2014*, pages 1280–1291. ACM Press, November 2014. doi:10.1145/2660267.2660291.

[Nec94] Vassiliy Ilyich Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2):165–172, February 1994. doi:10.1007/BF02113297.

[Nef01] C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In Michael K. Reiter and Pierangela Samarati, editors, *ACM CCS 2001*, pages 116–125. ACM Press, November 2001. doi:10.1145/501983.502000.

[OL] O1-Labs. Maller's Optimization to Reduce Proof Size - Mina Book. https://o1-labs.github.io/proof-systems/plonk/maller.html.

[PH78] S. Pohlig and M. Hellman. An improved algorithm for computing logarithms overgf(p)and its cryptographic significance (corresp.). *IEEE Transactions on Information Theory*, 24(1):106–110, 1978. doi:10.1109/TIT.1978.1055817.

[PHGR13] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE Computer Society Press, May 2013. doi:10.1109/SP.2013.47.

[PIK94] Choonsik Park, Kazutomo Itoh, and Kaoru Kurosawa. Efficient anonymous channel and all/nothing election scheme. In Tor Helleseth, editor, *EUROCRYPT'93*, volume 765 of *LNCS*, pages 248–259. Springer, Berlin, Heidelberg, May 1994. doi:10.1007/3-540-48285-7_21.

[Pol78] By J. M. Pollard. Monte carlo methods for index computation (mod p). *Mathematics of Computation*, 32(143):918–924, 1978.

[PR07] Manoj Prabhakaran and Mike Rosulek. Rerandomizable RCCA encryption. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 517–534. Springer, Berlin, Heidelberg, August 2007. doi:10.1007/978-3-540-74143-5_29.

[PR17] Olivier Pereira and Ronald L. Rivest. Marked mix-nets. In Michael Brenner, Kurt Rohloff, Joseph Bonneau, Andrew Miller, Peter Y. A. Ryan, Vanessa Teague, Andrea Bracciali, Massimiliano Sala, Federico Pintore, and Markus Jakobsson, editors, *FC 2017 Workshops*, volume 10323 of *LNCS*, pages 353–369. Springer, Cham, April 2017. doi:10.1007/978-3-319-70278-0_22.

[RMH+24] Michael Rosenberg, Tushar Mopuri, Hossein Hafezi, Ian Miers, and Pratyush Mishra. Hekaton: Horizontally-scalable zkSNARKs via proof aggregation. Cryptology ePrint Archive, Paper 2024/1208, 2024. URL: https://eprint.iacr.org/2024/1208.

[RS92] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 433–444. Springer, Berlin, Heidelberg, August 1992. doi:10.1007/3-540-46766-1_35.

[RS20] Lior Rotem and Gil Segev. Algebraic distinguishers: From discrete logarithms to decisional uber assumptions. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part III*, volume 12552 of *LNCS*, pages 366–389. Springer, Cham, November 2020. doi:10.1007/978-3-030-64381-2_13.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, February 1978. doi:10.1145/359340.359342.

[RZ21] Carla Ràfols and Arantxa Zapico. An algebraic framework for universal and updatable SNARKs. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 774–804, Virtual Event, August 2021. Springer, Cham. doi:10.1007/978-3-030-84242-0_27.

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th FOCS*, pages 543–553. IEEE Computer Society Press, October 1999. doi:10.1109/SFFCS.1999.814628.

[Set20] Srinath Setty. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 704–737. Springer, Cham, August 2020. doi:10.1007/978-3-030-56877-1_25.

[Sha49] Claude E. Shannon. Communication theory of secrecy systems. *Bell Systems Technical Journal*, 28(4):656–715, 1949.

[Sho97a] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997. doi:10.1137/S0097539795293172.

[Sho97b] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Berlin, Heidelberg, May 1997. doi:10.1007/3-540-69053-0_18.

[SK95] Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme - a practical solution to the implementation of a voting booth. In Louis C. Guillou and Jean-Jacques Quisquater, editors, *EUROCRYPT'95*, volume 921 of *LNCS*, pages 393–403. Springer, Berlin, Heidelberg, May 1995. doi:10.1007/3-540-49264-X_32.

[SL20] Srinath Setty and Jonathan Lee. Quarks: Quadruple-efficient transparent zkSNARKs. Cryptology ePrint Archive, Report 2020/1275, 2020. URL: https://eprint.iacr.org/2020/1275.

[STW24] Srinath T. V. Setty, Justin Thaler, and Riad S. Wahby. Unlocking the lookup singularity with Lasso. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part VI*, volume 14656 of *LNCS*, pages 180–209. Springer, Cham, May 2024. doi:10.1007/978-3-031-58751-1_7.

[Sze20] Alan Szepieniec. Polynomial IOPs for linear algebra relations. Cryptology ePrint Archive, Report 2020/1022, 2020. URL: https://eprint.iacr.org/2020/1022.

[TAB+20] Alin Tomescu, Ittai Abraham, Vitalik Buterin, Justin Drake, Dankrad Feist, and Dmitry Khovratovich. Aggregatable subvector commitments for stateless cryptocurrencies. In Clemente Galdi and Vladimir Kolesnikov, editors, *SCN 20*, volume 12238 of *LNCS*, pages 45–64. Springer, Cham, September 2020. doi:10.1007/978-3-030-57990-6_3.

[Tar75] Robert Endre Tarjan. Efficiency of a good but not linear set union algorithm. In *Journal of the ACM*, 1975. doi:10.1145/321879.321884.

[Tha13] Justin Thaler. Time-optimal interactive proofs for circuit evaluation. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 71–89. Springer, Berlin, Heidelberg, August 2013. doi:10.1007/978-3-642-40084-1_5.

[Tha22] Justin Thaler. FAQ on Jolt's initial implementation. https://a16zcrypto.com/posts/article/faqs-on-jolts-initial-implementation, April 2022.

[Tha24a] Justin Thaler. Accelerating the World Computer. https://a16zcrypto.com/posts/article/accelerating-the-world-computer-implementing-jolt/, April 2024.

[Tha24b] Justin Thaler. A New Era in SNARK Design. https://a16zcrypto.com/posts/article/a-new-era-in-snark-design-releasing-jolt/, April 2024.

[Tha24c] Justin Thaler. Understanding Jolt: Clarifications and reflections. https://a16zcrypto.com/posts/article/understanding-jolt-clarifications-and-reflections, June 2024.

[TW10] Björn Terelius and Douglas Wikström. Proofs of restricted shuffles. In Daniel J. Bernstein and Tanja Lange, editors, *AFRICACRYPT 10*, volume 6055 of *LNCS*, pages 100–113. Springer, Berlin, Heidelberg, May 2010. doi:10.1007/978-3-642-12678-9_7.

[TZ23] Stefano Tessaro and Chenzhi Zhu. Revisiting BBS signatures. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 691–721. Springer, Cham, April 2023. doi:10.1007/978-3-031-30589-4_24.

[WCY+21] Yi Wang, Rongmao Chen, Guomin Yang, Xinyi Huang, Baosheng Wang, and Moti Yung. Receiver-anonymity in rerandomizable RCCA-secure cryptosystems resolved. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 270–300, Virtual Event, August 2021. Springer, Cham. doi:10.1007/978-3-030-84259-8_10.

[Wee09] Hoeteck Wee. Zero knowledge in the random oracle model, revisited. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 417–434. Springer, Berlin, Heidelberg, December 2009. doi:10.1007/978-3-642-10366-7_25.

[Whi22] Barry Whitehat. Lookup singularity. https://zkresear.ch/t/lookup-singularity/65/7, December 2022.

[Wik05] Douglas Wikström. A sender verifiable mix-net and a new proof of a shuffle. In Bimal K. Roy, editor, *ASIACRYPT 2005*, volume 3788 of *LNCS*, pages 273–292. Springer, Berlin, Heidelberg, December 2005. doi:10.1007/11593447_15.

[Wik09] Douglas Wikström. A commitment-consistent proof of a shuffle. In Colin Boyd and Juan Manuel González Nieto, editors, *ACISP 09*, volume 5594 of *LNCS*, pages 407–421. Springer, Berlin, Heidelberg, July 2009. doi:10.1007/978-3-642-02620-1_28.

[Wik10] Douglas Wikström. Verificatum, 2010. https://www.verificatum.com.

[WTs+17] Riad S. Wahby, Ioanna Tzialla, abhi shelat, Justin Thaler, and Michael Walfish. Doubly-efficient zkSNARKs without trusted setup. Cryptology ePrint Archive, Report 2017/1132, 2017. URL: https://eprint.iacr.org/2017/1132.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986. doi:10.1109/SFCS.1986.25.

[ZBK+22] Arantxa Zapico, Vitalik Buterin, Dmitry Khovratovich, Mary Maller, Anca Nitulescu, and Mark Simkin. Caulk: Lookup arguments in sublinear time. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 3121–3134. ACM Press, November 2022. doi:10.1145/3548606.3560646.

[Zer] Risc Zero. Universal zero knowledge. https://risczero.com/.

[ZGK+18] Yupeng Zhang, Daniel Genkin, Jonathan Katz, Dimitrios Papadopoulos, and Charalampos Papamanthou. vRAM: Faster verifiable RAM with program-independent preprocessing. In *2018 IEEE Symposium on Security and Privacy*, pages 908–925. IEEE Computer Society Press, May 2018. doi:10.1109/SP.2018.00013.

[ZGK+22] Arantxa Zapico, Ariel Gabizon, Dmitry Khovratovich, Mary Maller, and Carla Ràfols. Baloo: Nearly optimal lookup arguments. Cryptology ePrint Archive, Report 2022/1565, 2022. URL: https://eprint.iacr.org/2022/1565.

[Zha22] Mark Zhandry. To label, or not to label (in generic groups). In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part III*, volume 13509 of *LNCS*, pages 66–96. Springer, Cham, August 2022. doi:10.1007/978-3-031-15982-4_3.

[Zip90] Richard Zippel. Interpolating polynomials from their values. *Journal of Symbolic Computation*, 9(3):375–403, 1990. Computational algebraic complexity editorial. URL: https://www.sciencedirect.com/science/article/pii/S0747717108800181, doi:10.1016/S0747-7171(08)80018-1.

[ZZ21] Mark Zhandry and Cong Zhang. The relationship between idealized models under computationally bounded adversaries. Cryptology ePrint Archive, Report 2021/240, 2021. URL: https://eprint.iacr.org/2021/240.