Outsourced Private Set Intersection for Pairwise Analytics

Ferran Alborch[1], Tangi De Kerdrel[2], Antonio Faonio[1], and Melek Önen[1]

EURECOM, Sophia Antipolis, France
{ferran.alborch,antonio.faonio,melek.onen}@eurecom.fr
Orange Innovation, Caen, France
tangi.dekerdrel@orange.com

Abstract. This paper studies privacy-preserving data analytics in settings where multiple parties hold sensitive datasets and want to compute global statistics without revealing their data. We focus on computing the total number of common elements (cardinality of intersections) across multiple pairs of datasets, while ensuring that only the final aggregated result is disclosed and no intermediate information (such as individual intersections) is leaked. To address this problem, we introduce a new cryptographic primitive called outsourced cardinality private set intersection with secret-shared outputs (CaOPSI-SS). Our solution is extremely simple and uses pseudorandom functions and two non-colluding servers to offload computation, making it suitable for environments with heterogeneous resources. Building on this primitive, we design a protocol for aggregated pairwise analytics that computes the sum of intersection cardinalities across many parties. We apply our framework to a real-world use case: privacy-preserving mail analytics in large organizations with multiple subsidiaries. The system allows useful fine-grained queries over email logs while protecting sensitive HR data. We also extend the solution with differential privacy mechanisms to further protect individual records. Finally, we implement and evaluate the protocol, showing its scalability and practicality for large datasets. Our solution enables parties to obliviously offload their datasets to two non-colluding servers using pseudorandom functions and further execute a circuit-PSI among these two servers to obtain secret shares of the output.

Keywords: Private Set Intersection · Delegated Computation · Multi-party Analytics.

1 Introduction

Performing large scale data analysis while keeping data privacy, has become a major concern ever since the exponential increase in the use of sensitive data in the digital world and the increasing need for compliance with regulations like GDPR. The objective of many stakeholders is that by computing some analytics over data, to obtain valuable insights to improve the quality of their products

or services without exposing sensitive information. The paradigm of privacy-preserving analytics then originated to allow parties to perform operations over data while keeping individual information secure.

In this work we consider the scenario where multiple entities, each owning a private dataset, want to compute aggregate statistics over some pairwise operation over the datasets. In particular, we are interested in the cardinality of the intersection, whicch has been used as a pairwise operation through multiple use cases including privacy-preserving contact tracing during the COVID-19 pandemic [36,5]. While there has been a lot of research on cardinality private set intersection [7,37], we investigate the scenario whereby a party is willing to compute the cardinality of the intersection among multiple pairs of datasets and obtain the ultimate sum (or aggregate cardinality), only. The main privacy goal in this scenario is that nothing more than the ultimate aggregate cardinality should be disclosed. In particular, when multiple datasets owned by different parties are involved in these analytics, the individual outputs of the pairwise operations should not be revealed to anyone. In a similar manner, the size of the datasets involved in the operation should not be leaked.

In addition, we will be considering a heterogeneous environment in terms of resources (computation and/or memory), where while some parties may have sufficient resources to store large databases and can perform costly computations at their side, some others may not. This consideration comes especially into play when the number of parties is large, since each party would have to perform the same operation with all the other parties, one-by-one. This may results in an unreasonable overhead for the parties with constrained resources.

To address this, we propose a new outsourced cardinality-PSI solution involving two non-colluding servers to perform the pairwise cardinality-PSI operations. The solution makes use of pseudorandom functions to enable parties send an oblivious version of their sets to the servers who later run a cardinality-PSI protocol where the output consists of secret shares of the cardinality, guaranteeing that the output of this protocol remains hidden. We further demonstrate the use of this primitive in a privacy-preserving mail analytics scenario whereby the two servers perform multiple, pairwise private set intersections over different mail databases and further output the sum of cardinalities as the final output, only. Finally, to further solidify the privacy preserving guarantees of the system, we also study the use of differential privacy mechanisms in order to protect the individual records in the private datasets.

Contributions. We propose the following contributions.

- **Outsourced private set intersection with hidden outputs:** We first define outsourced private set intersection with hidden outputs which we call CaOPSI-SS, for which we give an instantiation based on pseudo-random functions and cardinality circuit private set intersection as building blocks.
- **Outsourced analytics on pairwise intersections:** We then give an instantiation of outsourced analytics over pairwise intersections, which we call

PWPSI-Agg, based on outsourced private set intersection with hidden outputs.
– **Private mail analytics:** Finally, we use the pairwise analytics functionality to build a protocol for multi-entity mail analytics, considering both the standard as well as the differentially private case. This protocol is evaluated and fully benchmarked using a synthetic database generated based on mail traffic of a large corporate telecommunications group.

Outline. Section 2 reviews the state of the art. Section 3 defines the building blocks to construct our solution. In Section 4, we formally define outsourced cardinality PSI with hidden outputs and give a construction, whereas in Section5 we study the use case of mail analytics for a large corporate group making use of our new primitive. Finally, in Section 6 we analyze and benchmark a concrete implementation of our solutions

2 Related Work

Private Set Intersection (PSI) . PSI was first proposed by Meadows in [27] as a primitive capable of outputting the intersection of two hidden sets without leaking anything else, and instantiated through the Diffie-Hellman assumption. Since then, the field has been vastly studied, with instantiations from many different primitives, such as homomorphic encryption [9,8], polynomials over finite fields [20,15], or oblivious primitives such as oblivious transfer with oblivious pseudo-random functions [31,30] or oblivious key-value stores with vector oblivious linear evaluation [33,32].

Circuit-Private Set Intersection (cPSI). CPSI was first proposed by Huang et al. in [21] as a primitive where instead of outputting the elements in the intersection of two hidden sets, it outputs a circuit computed over such intersection and nothing else. More recently, the paradigm of cPSI constructions has veered towards PSI with secret shared outputs, where the protocols output a set of secret shares (one per element in one of the sets) with them being shares of 0 if the element is not in the intersection, or shares of 1 if the element is in the intersection [7,37]. This allows, indeed, to compute any arbitrary circuit over the intersection by leveraging generic multi-party computation paradigms [38,16,4].

Outsourced Private Set Intersection (OPSI). OPSI was proposed by Kerschbaum in [23,24] as a way to leverage the computational power of a third party to perform computationally expensive operations and, hence, allow less powerful parties to participate to the protocol. Similarly to standard PSI, OPSI has been instantiated from a plethora of primitives, such as homomorphic encryption [2,12], pseudo-random permutations [22,1], secret-sharing [28,26] or even proxy re-encryption [40,39]. For a more in-depth analysis and comparison we refer to the survey in [19].

In our quest to build a system capable of performing private mail analytics we are looking for three properties: outsourced computation, non-interactivity between subsidiary entities, and hidden outputs. While one can find a scheme satisfying any two of these three properties, to the best of our knowledge no scheme exists satisfying all three properties simultaneously. Therefore, we define a new primitive Cardinality Outsourced Private Set Intersection with secret-shared output (CaOPSI-SS), for which we give a construction based on cPSI, and then use to construct our protocol for Aggregated Pairwise Cardinality Private Set Intersection.

3 Preliminaries

3.1 Notation

For any $k \in \mathbb{Z}_{>0}$ we define $[k]$ as the set $\{1, \ldots, k\}$. Given any set $\mathcal{S}$, $s \leftarrow_R \mathcal{S}$ denotes that s is sampled uniformly at random in the set $\mathcal{S}$. Given a probability distribution D, we denote a sample y from D as $y \leftarrow D$. We denote probabilistic polynomial-time algorithms as PPT. Finally, we say a function f is *negligible* over n ($f = \text{negl}(n)$) if for all $k \in \mathbb{N}_{>0}$, there exists $n_0 \in \mathbb{N}_{>0}$ such that for any $n > n_0$, $|f(n)| < 1/n^k$.

3.2 Pseudo-Random Function

A pseudorandom function is a polynomial-time function that cannot be distinguished from a truly random function by an adversary. More formally, it is defined as follows:

Definition 1 (Pseudorandom function). *Let κ be a security parameter, K be a key space, X be an input space, Y be an output space, and $\mathcal{F}$ be the family of all functions from X to Y. We say that function $\Phi : K \times X \to Y$ is a* pseudorandom function (PRF) *if for any* PPT *distinguisher $\mathcal{A}$*

$$|Pr[k \leftarrow_R K : \mathcal{A}^{\Phi} = 1] - Pr[F \leftarrow_R \mathcal{F} : \mathcal{A}^{F} = 1]| \leq \text{negl}(\kappa).$$

3.3 Secret Sharing

A Secret sharing scheme [35] is a cryptographic primitive that allows n parties to share secret information in such a way that the secret can only be recovered when t of them cooperate. In this work, we make use of n-out-of-n secret sharing (where $n = t$) which is formally defined as follows.

Definition 2 (n-out-of-n Secret Sharing Scheme). *Let $\kappa \in \mathbb{Z}_{>0}$ be a security parameter. We define a* n-out-of-n secret sharing scheme *as the following tuple of* PPT *algorithms.*

- $\mathsf{SS.Share}(x)$*: given an element x as input, it outputs n secret shares $\{[\![x]\!]^{(i)}\}_{i \in [n]}$.*

Functionality 1 2-party Cardinality Circuit Private Set Intersection $\mathcal{F}_{\text{CacPSI}}$

Parameters. A secret sharing scheme $\mathsf{SS} = (\mathsf{Share}, \mathsf{Combine})$.
Inputs. Party P_1 inputs a set X_1 of elements, P_2 inputs a set X_2 of elements.
Outputs. P_1 and P_2 output $[\![c]\!]^{(i)}$ for $i \in \{1, 2\}$ secret shares of $c = |X_1 \cap X_2|$.

– $\mathsf{SS.Combine}(\{[\![x]\!]^{(i)}\}_{i\in[n]})$*: given* n *compatible secret shares* $\{[\![x]\!]^{(i)}\}_{i\in[n]}$*, it outputs a string* res.

There are many different ways of instantiating secret sharing schemes, each with their own properties. In this work we will focus on linear secret sharing schemes, which have the property of being linearly homomorphic. In other words for any coefficients a, b and elements x_0, x_1, we have

$$a \cdot [\![x_0]\!]^{(i)} + b \cdot [\![x_1]\!]^{(i)} = [\![a \cdot x_0 + b \cdot x_1]\!]^{(i)},$$

for $i \in \{1, 2\}$.

3.4 Private Set Intersection

Private Set Intersection (PSI) is a multi-party protocol allowing n different parties $P_1, \ldots, P_n$ owning respective private sets $X_1, \ldots, X_n$ to obliviously compute the intersection $X_1 \cap \ldots \cap X_n$ without revealing any other information about the private sets. Since the first proposals, many different variants have been proposed and studied to handle differing particular problems. Relevant to this work, we will deal with 2-party circuit-PSI and outsourced PSI protocols.

Circuit PSI. A circuit-PSI (cPSI) protocols is a generalization of PSI protocol where instead of outputting the intersection in clear, it outputs secret shares of the intersection. Then, generic 2-party computation [38,16,4] can be used to compute any arbitrary circuit over the intersection. More specifically, parties P_1, P_2 input sets X_1, X_2 respectively, and for each element in X_1, the protocol outputs a secret share of 1 if the element is in $X_1 \cap X_2$ and a secret share of 0 otherwise.

In this work, we use a special case called cardinality circuit PSI, where the output a single secret share of the cardinality of the intersection instead of a secret share for each element. For the ideal functionality see Functionality 1. The ideal functionality is parameterized by the secret sharing scheme.

Remark 1. Note that cardinality-circuit-PSI is trivially constructible from standard circuit-PSI (where the outputs are secret shares of 1 or 0 depending on whether an element is in the intersection or not) with linear secret shares. To construct the shares of the cardinal, each party locally aggregates all the secret shares corresponding to each element.

Functionality 2 $\mathcal{F}_{\text{CaOPSI-SS}}$

Inputs. Party P_1 inputs a set X_1 of elements, P_2 inputs a set X_2 of elements, the servers S_1, S_2 input nothing.
Outputs. P_1 and P_2 output nothing, S_1 and S_2 output $[\![c]\!]^{(i)}$ for $i \in \{1, 2\}$ secret shares of $c = |X_1 \cap X_2|$.
Leakage. If a server S_i is corrupted, the adversary receives $|X_1|$ and $|X_2|$.

4 Construction for Pairwise Analytics: From CacPSI

4.1 Outsourced PSI with Hidden Outputs

An outsourced (or delegated) PSI protocols is a variant of standard PSI protocol where computationally expensive operations are outsourced powerful but untrusted servers. An outsourced PSI may differ depending on how involved the servers are in the protocol, whether it is the servers or the parties who store the data and whether it is the servers or the parties who obtain the result after the computation is done.

In this work, the clients store the data while the servers receives the output, which is a secret sharing of the cardinality. We call this functionality CaOPSI-SS. More specifically, parties P_1, P_2 input X_1, X_2 respectively and the servers S_1, S_2 input nothing, and the protocol outputs to the server S_i the secret share $[\![c]\!]^{(i)}$, which are secret shares of $c = |X_1 \cap X_2|$ and nothing to P_1, P_2. For the ideal functionality see Functionality 2.

Remark 2. As in Remark 1, one can define outsourced PSI with shared outputs as releasing a secret share of either 1 or 0 per element in one of the sets, which can be trivially transformed into sharing the cardinality.

Protocol Description. The protocol is described in Protocol 1. At a high level, it is quite simple. Each party encodes its input set using a pseudorandom function; the resulting pseudorandom values are then sent to the servers, which execute a standard cardinality PSI protocol. The protocol is outsourced in the sense that the parties perform only a linear number (in the size of their sets) of symmetric-key operations.

Correctness and Security.

Theorem 1. *Assume that Φ is a pseudorandom function. Then the protocol $\Pi_{CaOPSI\text{-}SS}$ correctly and securely realizes the ideal functionality $\mathcal{F}_{CaOPSI\text{-}SS}$ in the $\mathcal{F}_{CaPSI}$-hybrid model in the presence of a single semi-honest adversary.*

Proof Sketch. We provide a proof sketch. Correctness follows from the fact that, except with negligible probability due to collisions in the PRF outputs, the

Protocol 1 $\Pi_{\text{CaOPSI-SS}}$

Inputs: Party P_1 inputs a set X_1 of elements, P_2 inputs a set X_2 of elements, the servers S_1, S_2 input nothing.
Outputs: P_1 and P_2 output nothing, S_1 and S_2 output $[\![c]\!]^{(i)}$ for $i \in \{1,2\}$ secret shares of $c = |X_1 \cap X_2|$.

The protocol:
Offline Step, parties execute only once:

0. Each party P_i samples $k_i \leftarrow \{0,1\}^n$ and sends k_i to the other party P_{2-i}. The parties set $k \leftarrow k_1 \oplus k_2$.

Online Steps:

1. Each party P_i computes the elements $\tilde{X}_i := \Phi_k(X_i)$ and sends them to S_i. Where $\tilde{X}_i := \{\Phi_k(x_{ij}) : x_{ij} \in X_i\}$.
2. S_1 and S_2 run $([\![c]\!]^{(1)}; [\![c]\!]^{(2)}) \leftarrow \mathcal{F}_{\text{CaPSI}}(\widetilde{X}_1; \widetilde{X}_2)$, and output $[\![c]\!]^{(i)}$ respectively.

servers correctly execute $\mathcal{F}_{\text{CaPSI}}$. Since the PRF is evaluated under a uniformly random secret key, the probability of such collisions is negligible.

The simulation for a corrupted party P_i is immediate, as parties receive no output and send only a single message to their respective server.

If instead a server is corrupted, say S_1, the simulator generates the message from P_1 to S_1 by sending $|X_1|$ uniformly random strings. It then provides S_1 with its output share, as specified by the ideal functionality. Any distinguisher for S_1 that can distinguish the simulated message from the real one can be used to break the pseudorandomness of Φ. □

4.2 Analytics over Pairwise Intersections

We define a new cryptographic primitive called aggregated pairwise cardinality private set intersection, PWPSI-Aggin short. Briefly, this is a functionality with n clients and two servers. The input provided by client P_i is a list of $n-1$ sets $(X_j^i)_{j\in[n],j\neq i}$, while the servers do not provide any input the functionality computes computes $s = \sum_{i<j} |X_j^i \cap X_i^j|$, which is secret shared and sent to the two servers.

Protocol Description. The protocol for analytics over pairwise intersections uses as a building block a protocol for outsourced cardinality PSI parametrized with a linear secret sharing scheme over $\mathbb{Z}_q$ for $q > 2^n$. The idea of the protocol is rather easy. For each couple P_i, P_j they run an $\Pi_{\text{CaOPSI-SS}}$ with the servers S_1 and S_2 where the clients P_i, P_j inputs the sets X_j^i, X_i^j, such a protocol execution

Functionality 3 $\mathcal{F}_{\text{PWPSI-Agg}}$

Inputs. Each party P_i for $i \in [n]$ inputs a set of sets of elements $X^i = \{X^i_j\}_{j \neq i}$, the servers S_1, S_2 input nothing.
Outputs. The parties P_i output nothing, the servers S_1, S_2 output a secret sharing $[\![s]\!]$ where:

$$s = \sum_{i<j}^{n} |X^i_j \cap X^j_i|.$$

Leakage. If a server S_i is corrupted, the adversary receives $|X^i_j|$ for any i, j.

outputs a secret share $[\![c^i_j]\!]$ to the servers. At the end, the servers aggregate all the outputs computing $\sum_{i,j}[\![c^i_j]\!]$. The formal description is in Protocol 2.

Correctness and Security. We prove correctness and security of $\Pi_{\text{PWPSI-Agg}}$.

Theorem 2. *Protocol $\Pi_{PWPSI\text{-}Agg}$ presented in Protocol 2 correctly and securely realizes the ideal functionality $\mathcal{F}_{PWPSI\text{-}Agg}$ in the $\mathcal{F}_{CaOPSI\text{-}SS}$-hybrid model against a semi-honest adversary, assuming that the linear secret sharing scheme is correct and holds secrecy.*

Proof. Let us start with correctness. Due to the correctness of $\mathcal{F}_{\text{CaOPSI-SS}}$ we know that $[\![c^i_j]\!]^{(1)}, [\![c^i_j]\!]^{(2)}$ are 2-out-of-2 linear secret shares of $|X^i_j \cap X^j_i|$. Then, due to the linearity of the secret sharing scheme we know that

$$[\![s]\!] = \textstyle\sum_{i<j}[\![c^i_j]\!] = \left[\!\!\left[\sum_{i<j}^{n} c^i_j\right]\!\!\right] = [\![\sum_{i<j}^{n} |X^i_j \cap X^j_i|]\!].$$

For security, the simulator description is straightforward, as the protocol only invokes the ideal functionality $\mathcal{F}_{\text{CaOPSI-SS}}$.

If one of the servers S_1, S_2 is corrupted, say S_1, the simulator must generate the partial shares $[\![c^i_j]\!]^{(1)}$ for all $i < j$. We notice that all shares $[\![c^i_j]\!]$ can be set to shares of 0. By the privacy of the secret sharing scheme, the simulated and real views are indistinguishable.

We remark that the security argument covers the case where one of the parties in $\{P_1, \ldots, P_n, S_1, S_2\}$ is corrupted and S is corrupted. Thus, the access structure in the corruption model differs slightly from the one stated (as it is more general). However, this discrepancy can be resolved by identifying S with S_1, matching the corruption model assumed in the theorem.

□

5 Construction for Mail Analytics

In this section, we show how to apply PWPSI-Agg to the use case of privacy-preserving mail analytics in a large corporate group. We first formalize the class

Protocol 2 $\Pi_{\text{PWPSI-Agg}}$

Inputs: Each party P_i inputs a set of $n-1$ sets of elements $X^i = \{X_j^i\}_{j \neq i}$.
Outputs: Each party P_i outputs nothing. The server S outputs

$$s = \sum_{i<j}^{n} |X_j^i \cap X_i^j|.$$

The protocol:

1. For every i, j with $i < j$, the parties P_i and P_j send respectively X_j^i and X_i^j to the $\mathcal{F}_{\text{CaOPSI-SS}}$, the servers receive secret shares $[\![c_j^i]\!]$ from the ideal functionality.
2. Each server S_k for $k \in \{1, 2\}$ locally computes

$$[\![s]\!]^{(k)} = \sum_{i<j}^{n} [\![c_j^i]\!]^{(k)},$$

and output their shares.

of queries, and then describe how to evaluate them using $\mathcal{F}_{\text{PWPSI-Agg}}$. We present two variants of the protocol. The second variant additionally incorporates differential privacy [11].

5.1 Defining the Use Case

Consider the use case of a large corporate telecommunications group, let us call it Tangerine S.A., wanting to perform mail analytics over their work-force sensitive HR information. Notably, the workers from Tangerine S.A. are divided into different subsidiaries, which are independent juridic entities (for example, being based in different countries). This means that the HR information of the workers is stored by their respective subsidiaries, and more importantly, due to privacy regulations such as GDPR, they cannot be divulged in clear to any other entities.

This setting severely complicates performing useful data analytics over email flow by the headquarters of Tangerine S.A. (which for the purpose of this work we will consider also as a separate juridic entity). For example, queries like how many mails were sent from workers located on one time zone to another during a certain time frame , the amount of mails sent from product manager to developers in different locations , or the average size of mails sent from upper hierarchy to the workers , become significantly more difficult when taking into account mails between different subsidiaries.

More specifically, we assume a set of subsidiaries (or independent legal entities) with each having access to two databases. The first one, is the private

ID	Gender	Location	Job Title	Workday	⋯
1	F	Silicon Valley	Product Manager	40 hpw	⋯
2	M	Paris	Developer	35 hpw	⋯
3	Other	Paris	Administrative	35 hpw	⋯
4	M	Brussels	Developer	40 hpw	⋯
⋮	⋮	⋮	⋮	⋮	⋮

(a) Private database with HR data.

Sender	S-Entity	Receiver	R-Entity	Metadata
7	A	38	F	M_1
3	C	21	A	M_2
54	A	15	A	M_3
47	A	2	D	M_4
⋮	⋮	⋮	⋮	⋮

(b) Mail log.

Fig. 1: Databases stored by subsidiary "A".

database containing the HR information of their employees such as their gender, location, job title or workday weekly hours. The second one is a mail log, storing all mails either sent or received (or both) by an employee of that subsidiary, including the ID and entity of both the sender and receiver as well as the metadata of the mail. For an example of these databases, see Figure 1. In this context, given a query by the company headquarters, the expected result would be the sum of the cardinalities of the pairwise intersections of mail log databases filtered appropriately with the respective private database through the policies given by the query.

The previously proposed protocol for analytics over pairwise intersections is especially suitable for this use case because:

- The number of mails exchanged daily may count in the hundreds of thousands. Thus a solution involving third party-servers like in PWPSI-Agg is scalable with respect to the size of the databases;
- The proposed PWPSI-Agg solution ensures that the partial cardinality results remain hidden. Disclosing these partial intersections can be a severe privacy breach, for example in the case where some subsidiaries are small in the number of employees.
- Tangerine S.A. is a multinational company, thus different subsidiaries might have different office hours, since they are located in different geographical areas. Therefore, outsourcing computation to two servers and minimizing interaction among subsidiaries would definitely help in this context.

Remark 3. Despite the concreteness of this use case, the setup is equally applicable to other scenarios where counting interactions between entities with respect to different private filters is needed. For example, this same setup works for data analytics over inter-entity bank transfers. In this situation the private database will hold the confidential information of the clients while the logs will contain the bank transfers. Then, controlling entities can privately analyze the transfer flow without compromising the confidential information of the clients.

5.2 Pairwise Analytics with Additional Local Inputs

In addition to the result of common emails among pairwise subsidiaries, Tangerine S.A. wishes to also consider the internal emails (emails sent between

workers of the same entity). To deal with this additional feature, each party inputs an extra integer value v_i, and each server outputs the sum of all these values v_i together with the cardinality of the pairwise intersections, i.e., $s = \sum_{i=1}^{n} v_i + \sum_{i<j}^{n} |X_j^i \cap X_i^j|$. To account for this, in the protocol, during step 1 every client generates 2-out-of-2 linear secret shares $[\![v_i]\!]^{(1)}, [\![v_i]\!]^{(2)} \leftarrow \mathsf{SS.Share}(v_i)$, and sends to server S_i their respective share. Then, at step 2, the servers compute $[\![s]\!]^{(k)} = \sum_{i=1}^{n} [\![v_i]\!]^{(k)} + \sum_{i<j}^{n} [\![c_j^i]\!]^{(k)}$.

5.3 Formal Definition of Query

We assume that the mail logs contain, for every mail, the sender's and receiver's entity ID, the sender's and receiver's worker ID and the metadata of the mail. The HR database contains, for every worker, their ID and k different definitional fields (i.e. gender, location, job title, workday...). The queries give a time frame from which to count mails as well as a subset of entities to focus in, and then policies for each of the sender's and receiver's respective fields. In other words:

> *In time period T, how many mails are sent between a given subset E of entities, from workers satisfying policy Pol_i^s for every field $i \in [k]$ and received by workers satisfying policy Pol_i^r for every field $i \in [k]$.*

With this characterization, we can say that any query is determined by the tuple $q = (T, E, \{\mathrm{Pol}_i^s, \mathrm{Pol}_i^r\}_{i\in[k]})$. The exact answer to such a query q is the following integer value:

$$\sum_{i,j\in E} \left| \left\{ m : \begin{array}{c} \wedge_k \mathrm{Pol}_k^s(m.\texttt{sender}),\ m \in X_j^i \\ m.\texttt{time} \in T \end{array} \right\} \cap \left\{ m : \begin{array}{c} \wedge_k \mathrm{Pol}_k^r(m.\texttt{receiver}),\ m \in X_i^j \\ m.\texttt{time} \in T \end{array} \right\} \right|$$

5.4 Instantiation from PWPSI-Agg

We explain how to properly build the sets of elements $\{X_j^i\}_{j\neq i}$ as well as the values v_i so that the output of $\mathcal{F}_{\text{PWPSI-Agg}}$ is the desired query applied to the mail logs and private HR databases. We discuss both cases where the analysis is desired with differential privacy and without it.

Without DP. The sets $\{X_j^i\}_{j\neq i}$ are the mails sent between entities P_i and P_j and satisfy the policies from the side of P_i, while v_i corresponds to the internal mails in party P_i satisfying the policies. More in details, first P_i goes through its HR database, and creates a list of valid $\mathsf{senders}$ and $\mathsf{receivers}$ using $\{\mathrm{Pol}_\ell^s, \mathrm{Pol}_\ell^r\}_{\ell\in[k]}$. Then, it goes through its mail logs, and for each internal email, if both sender and receiver are in the lists, then the internal mail counter v_i is incremented by one. Finally, for each external email between P_i and P_j, if the sender is in the list $\mathsf{senders}$ (analogously in the case of the receiver) then the mail gets added to X_j^i. This data treatment together with $\mathcal{F}_{\text{PWPSI-Agg}}$ gives the

Algorithm 1 Precomputation for party i. Parts in blue refer to steps unique to the DP solution.

Input: Databases $\{\mathrm{ML}_i, \mathrm{HR}_i\}_{i\in[n]}$ referring to the mail log and private HR database respectively, query $q = (T, E, \{\mathrm{Pol}^s_j, \mathrm{Pol}^r_j\}_{j\in[k]})$, privacy parameters: privacy budget ε, sensitivity Δ (maximum number of mails sent per worker)

Output: Output $v_i, \{X^i_j\}_{j\neq i}$

```
if i ∉ E then                                   ▷ If party is not involved in the query abort
    Abort
Set senders = receivers = ∅
for all worker ∈ HR_i do                        ▷ Choose workers satisfying policy
    Add worker to senders and receivers
    for all j ∈ [k] do
        if field j does not satisfy Pol^s_j then
            Remove worker from senders
        if field j does not satisfy Pol^r_j then
            Remove worker from receivers
Set v_i = 0, X^i_j = ∅ for all j ≠ i
for all j ∈ [n], j ≠ i do                       ▷ Build set X^i_j and internal emails
    Set internal = 0
    for all mail ∈ ML_i and time ∈ T do
        if sender entity = receiver entity = i then      ▷ Count internal mails
            if sender ∈ senders and receiver ∈ receivers then
                internal = internal + 1
        else                                             ▷ Build X^i_j for all j ≠ i
            if sender entity = i and sender ∈ senders then
                Add mail to X^i_j
            else if receiver entity = i and receiver ∈ receivers then
                Add mail to X^i_j
    v_i = v_i + internal
Let k = |E|                                     ▷ Add the DP noise
Sample z_i ← SGDL(1/k, exp(−ε/Δ))
v_i = v_i + z_i
```

correct result, since for every email to be counted (i.e. email satisfying both sender and receiver properties), if it is an internal email it is counted in v_i, and otherwise it appears in the intersection $X^i_j \cap X^j_i$ for some $j \neq i$. For a formal description of this , see Algorithm 1.

With DP. Note that the sets $\{X^i_j\}_{j\neq i}$ and internal emails are computed as in the previous paragraph. However, we need to add some noise distribution to v_i to satisfy differential privacy. More in details, let k be the number of entities involved in the query (i.e. $|E|$), then P_i samples $z_i \leftarrow \mathsf{SGDL}(1/k, \exp(-\varepsilon/\Delta))$ where Δ is the maximum number of mails sent per worker, and updates v_i by adding z_i to it. This gives us the desired DP result, since the output of $\mathcal{F}_{\text{PWPSI-Agg}}$ gives the desired query result (analogously to the previous paragraph) plus $\sum_{i\in E} z_i$. Then, referring to Appendix A, by Lemma 1 we know that $\sum_{i\in E} z_i \sim \mathsf{Geo}(\exp(-\varepsilon/\Delta))$, which by Lemma 2 we know it is an $(\varepsilon, 0)$-DP mech-

anism. As such, we achieve a DP compliant protocol with negligible overcost, with the only trade-off being in utility.

Remark 4. In this work we focus on feasibility, we use the most standard DP mechanism for our scenario, which is the local geometric mechanism. The utility could be improved by leveraging certain trade-offs depending on the specific scenario, such as a trusted shuffler [10], prior knowledge about the distribution of the data [6,3] or more accurate knowledge on the sensitivity of each query [29].

Remark 5. One last open question is the way to choose the sensitivity Δ. This parameter (only involved for the computations involving DP) is assumed to be the maximum possible mails sent by one worker in a day. This, however, may be difficult to parametrize in a real life scenario where some outliers may send a substantially greater than the mean, having a significant impact on the utility of the scheme. To avoid this, we can use the fact that the geometric mechanism satisfies $(\varepsilon, 0)$-DP. Then, by setting Δ to be the maximum for *most* of the employees, the outliers are covered by δ in (ε, δ)-DP.

6 Evaluation

6.1 Experimental setting

We have implemented[1] our construction of PWPSI-Agg for mail analytics in Rust. We use the CPSI construction in [7] because of its high efficiency (both asymptotically and concretely) and the existing two versions demonstrating its flexibility in performance. To transform this solution into a CacPSI construction we perform the standard transformation of swapping to arithmetic secret shares and then locally add them all together (see Remark 1). To implement these protocols we use the `swanky` library [13] for oblivious transfer and oblivious transfer extension, ands the `voprf` library [25] for oblivious pseudo-random functions.

Experiments are executed on a Intel Core Ultra 9 285K (24C/24T, 6.5 GHz, AVX2), with 248 GB of RAM running Ubuntu 24.04 operating system, where each auxiliary server S_i is assigned 12 threads, and therefore up to 12 CaOPSI-SS are run, concurrently.

6.2 Performance Analysis

Scalability Experiment. The first analysis consists of the scalability of our construction with respect to the number of entities on the one side and the number of employees per entity, on the other side. The query involves all entities and the policy for senders (as well as for receivers) is sampled at random by taking 75% of them. For the scalability experiment with respect to the number of entities, we fix the number of employees per entity to 30 and vary the number

[1] Code available in open-source upon acceptance.

# Entities	Offline time (s)	Online time (s) (Sequential)	# pairs	Online time (s) (Parallel)	Comm. (KB)
3	0.71	89.28	3	35.16	4.07
4	1.367	130.85	6	27.74	6.84
5	2.20	161.95	10	23.12	10.09
6	3.68	202.68	15	18.68	11.83
7	5.56	254.47	21	18.39	15.41

(a) Scaling with respect to entities. All entities have 30 employees.

Employees per entity	Offline time (s)	Online time (s) (Sequential)	Online time (s) (Parallel)	Comm. (KB)
30	0.75	95.20	36.93	4.37
40	0.72	90.05	33.96	4.25
50	0.74	98.74	38.51	4.25
60	0.69	96.95	40.67	4.18
70	0.74	86.70	33.54	4.53

(b) Scaling with respect to number of employees per entity. All experiments are with 3 entities.

Table 1: Scalability experiment for our mail solution.

of entities, while for the scalability experiment with respect to the number of employees per entity, we fix the number of entities to 3 and vary the over the number of employees. This setting ensures that in both experiments the total number of employees remains the same at the same row in Tables 1a and 1b.

In Table 1 shows the performance results with respect to these two experiments. The sequential online time is evaluated assuming that all executions of CaOPSI-SS are performed one after the other and the parallelized online time is evaluated assuming an idealized scenario where all CaOPSI-SS executions can be run in parallel (i.e. it shows the slowest individual CaOPSI-SS execution). As expected, we observe that the sequential online computational time for PWPSI-Agg is much more sensitive to the increasing of the number of equal-sized entities than to the increasing of the amount of employees per entity. However, we notice a steady decrease in the parallel computational time. This is due to the fact that as the number of entities increase, the number of emails exchanged pairwise decreases, since the number of pairs increases quadratically while the amount of mails only increases linearly. Consequently, the growth we see in online computational time for PWPSI-Agg is sub-quadratic.

Additionally, we also consider two well-known scenarios: cross-silo (where few very large and computationally powerful entities interact) and cross-device (where many small and computationally-constrained entities interact). Applied to our use-case, we define 15,000 employees, which are distributed to 3 entities in the cross-silo scenario, and to 100 entities in the cross-device one. In Table 2, we remark that the cross-device scenario incurs better computation time since the size of the pairwise sets tends to 0 whereas the cross-silo scenario becomes more

Scenario	Offline time (s)	Online time (s) (Sequential)	# pairs	Online time (s) (Parallel)	Comm. (KB)
Cross-silo	2.57	2 829.69	3	1 012.24	59.91
Cross-device	1 351.47	609.74	4 950†	7.39	1 360

†Note that most pairs have empty sets and as such are not executed.

Table 2: Cross-silo versus cross-device comparison.

Offline time (s)	Online time (s) (Sequential)	# pairs	Online time (s) (Parallel)	Comm. (KB)
1 306.73	776.61	4 950†	44.82	11 930

†Note that most pairs have empty sets and as such are not executed.

Table 3: Experiments over a realistic synthetic dataset.

efficient in terms of communication cost. This is due to the increased advantage of oblivious transfer extension when more oblivious transfers are needed.

Realistic Experiment. The second experiment aims at evaluating the feasibility of the solution using a synthetic dataset modeling the traffic of a large telecommunication corporate group (for more details on the generation of this dataset see Appendix C). In particular, we analyze a more realistic policy for the query: in particular, we randomly sample at 10%, each possible field for both receiver and sender. The performance results are shown in Table 3. We observe that more than half of the execution time is reserved for offline computation, corresponding to the setup for all the oblivious transfer extensions between S_1 and S_2 and this can easily be performed before-hand. The online-time, on the other hand is measured as 10 minutes which seems reasonable for a privacy-preserving query over mail traffic scenario in a corporate group with over 100,000 workers.

7 Conclusion

This work leaves the door open to interesting future research. For example, a construction of PWPSI-Agg without the use of auxiliary servers is an interesting open problem, as well as dealing with the malicious security case. On the more practical side, PWPSI-Agg does not provide a framework to perform other data analytics than counting, for example, analysis over the size of the mails is left as a difficult open problem. Finally, at the implementation level, due to the disparity in sizes of databases (few big entities and a lot of small ones) a mix of balanced CaCPSI architecture (where the cost grows linearly with the size of both sets) like [7] and unbalanced CaCPSI architecture (where the cost grows linearly with the size of the small set and sublinearly with the size of the big set) like [18] would be ideal, as well as adding parallelization at the CaCPSI level.

References

1. Abadi, A., Dong, C., Murdoch, S.J., Terzis, S.: Multi-party updatable delegated private set intersection. In: Financial Cryptography and Data Security. pp. 100–119 (2022)
2. Abadi, A., Terzis, S., Dong, C.: O-psi: Delegated private set intersection on outsourced datasets. In: ICT Systems Security and Privacy Protection. pp. 3–17 (2015)
3. Alborch, F., Athanasiou, A., Reisert, P.: Optimizing Differential Privacy in Federated Analytics under Known Input Distributions . In: IEEE Computer Security Foundations Symposium. CSF '26 (2026)
4. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: ACM Symposium on Theory of Computing. pp. 1–10. STOC '88 (1988)
5. Berke, A., Bakker, M., Vepakomma, P., Larson, K., Pentland, A.S.: Assessing disease exposure risk with location data: A proposal for cryptographic preservation of privacy (2020)
6. Bhaskar, R., Bhowmick, A., Goyal, V., Laxman, S., Thakurta, A.: Noiseless database privacy. In: ASIACRYPT. pp. 215–232 (2011)
7. Chandran, N., Gupta, D., Shah, A.: Circuit-psi with linear complexity via relaxed batch opprf. Privacy Enhancing Technologies Symposium **2022**(1), 353–372 (2022)
8. Chen, H., Huang, Z., Laine, K., Rindal, P.: Labeled psi from fully homomorphic encryption with malicious security. In: ACM Conference on Computer and Communications Security. p. 1223–1237. CCS '18 (2018)
9. Chen, H., Laine, K., Rindal, P.: Fast private set intersection from homomorphic encryption. In: ACM Conference on Computer and Communications Security. p. 1243–1255. CCS '17 (2017)
10. Cheu, A., Smith, A., Ullman, J., Zeber, D., Zhilyaev, M.: Distributed differential privacy via shuffling. In: EUROCRYPT. pp. 375–403 (2019)
11. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: Theory of Cryptography. pp. 265–284. TCC '06 (2006)
12. Fan, C., Jia, P., Lin, M., Wei, L., Guo, P., Zhao, X., Liu, X.: Cloud-assisted private set intersection via multi-key fully homomorphic encryption. Mathematics **11**(8) (2023)
13. Galois, Inc.: swanky: A suite of rust libraries for secure computation. https://github.com/GaloisInc/swanky (2019)
14. Ghosh, A., Roughgarden, T., Sundararajan, M.: Universally Utility-Maximizing Privacy Mechanisms. In: ACM Symposium on Theory of Computing. pp. 351–360. STOC '09 (2009)
15. Ghosh, S., Nilges, T.: An algebraic approach to maliciously secure private set intersection. In: EUROCRYPT. pp. 154–185 (2019)
16. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game. In: ACM Symposium on Theory of Computing. pp. 218–229. STOC '87 (1987)
17. Goryczka, S., Xiong, L.: A comprehensive comparison of multiparty secure additions with differential privacy. IEEE Transactions on Dependable and Secure Computing **14**(5), 463–477 (2017)
18. Hao, M., Liu, W., Peng, L., Li, H., Zhang, C., Chen, H., Zhang, T.: Unbalanced Circuit-PSI from oblivious Key-Value retrieval. In: USENIX Security Symposium. pp. 6435–6451 (2024)
19. Hawkes, S., Weinert, C.: Sok: Outsourced private set intersection. In: Applied Cryptography and Network Security. ACNS '26' (2026)

20. Hazay, C.: Oblivious polynomial evaluation and secure set-intersection from algebraic prfs. In: Theory of Cryptography. pp. 90–120. TCC '15 (2015)
21. Huang, Y., Evans, D., Katz, J.: Private set intersection: Are garbled circuits better than custom protocols? In: Network and Distributed System Security Symposium. NDSS '12 (2012)
22. Kamara, S., Mohassel, P., Raykova, M., Sadeghian, S.S.: Scaling private set intersection to billion-element sets. In: Financial Cryptography and Data Security. pp. 195–215 (2014)
23. Kerschbaum, F.: Collusion-resistant outsourcing of private set intersection. In: ACM Symposium on Applied Computing. p. 1451–1456. SAC '12 (2012)
24. Kerschbaum, F.: Outsourced private set intersection using homomorphic encryption. In: ACM Symposium on Information, Computer and Communications Security. p. 85–86. ASIACCS '12 (2012)
25. Lewi, K.: voprf. https://github.com/facebook/voprf (2021)
26. Li, Y., Ghosh, D., Gupta, P., Mehrotra, S., Panwar, N., Sharma, S.: Prism: Private verifiable set computation over multi-owner outsourced databases. In: International Conference on Management of Data. p. 1116–1128. SIGMOD '21 (2021)
27. Meadows, C.: A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. In: IEEE Symposium on Security and Privacy. pp. 134–134. S&P '86 (1986)
28. Miyaji, A., Nakasho, K., Nishida, S.: Privacy-preserving integration of medical data. J. Med. Syst. **41**(3), 1–10 (2017)
29. Nissim, K., Raskhodnikova, S., Smith, A.: Smooth sensitivity and sampling in private data analysis. In: Symposium on Theory of Computing. pp. 75–84. STOC '07 (2007)
30. Pinkas, B., Rosulek, M., Trieu, N., Yanai, A.: Spot-light: Lightweight private set intersection from sparse ot extension. In: CRYPTO. pp. 401–431 (2019)
31. Pinkas, B., Schneider, T., Zohner, M.: Faster private set intersection based on ot extension. In: USENIX Security Symposium. pp. 797–812 (2014)
32. Raghuraman, S., Rindal, P.: Blazing fast psi from improved okvs and subfield vole. In: ACM Conference on Computer and Communications Security. p. 2505–2517. CCS '22 (2022)
33. Rindal, P., Schoppmann, P.: Vole-psi: Fast oprf and circuit-psi from vector-ole. In: EUROCRYPT. pp. 901–930 (2021)
34. Seetha Lekshmi, V., Sebastian, S.: A skewed generalized discrete laplace distribution. Int J Math Stat Invent **2**, 95–102 (2014)
35. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979)
36. Trieu, N., Shehata, K., Saxena, P., Shokri, R., Song, D.: Epione: Lightweight contact tracing with strong privacy (2020)
37. Yang, Y., Liang, X., Song, X., Dong, Y., Huang, L., Ren, H., Dong, C., Zhou, J.: Maliciously secure circuit private set intersection via spdz-compatible oblivious PRF. Privacy Enhancing Technologies Symposium **2025**(2), 680–696 (2025)
38. Yao, A.C.C.: How to generate and exchange secrets. In: Symposium on Foundations of Computer Science. pp. 162–167. SFCS '86 (1986)
39. Zhang, E., Li, F., Niu, B., Wang, Y.: Server-aided private set intersection based on reputation. Inf. Sci. **387**(C), 180–194 (2017)
40. Zheng, Q., Xu, S.: Verifiable delegated set intersection operations on outsourced encrypted data. In: IEEE International Conference on Cloud Engineering. pp. 175–184 (2015)

A Preliminaries on Differential Privacy

Differential Privacy [11] is a privacy notion that aims to accurately measure the trade-off between guaranteeing the privacy of the data of an individual inside a (large) database and the utility of the statistics obtained from such a private data analysis. More particularly, it is based on the concept of *neighbouring databases*, which refer to two databases X, X' that differ in one record, only, and it analyzes *privacy mechanisms*, which are probabilistic functions $\mathcal{M}$ applied to the databases with output in the multi-dimensional real numbers. The formal definition goes as follows.

Definition 3 (Differential Privacy). *Let ε, δ be two positive real values. We say a mechanism $\mathcal{M}$ satisfies* (ε, δ)-differentially privacy *(DP) if for all neighbouring databases X, X' and any set $S \subset \mathbb{R}^k$ the following holds true* $\Pr[\mathcal{M}(X) \in S \leq \exp(\varepsilon) \cdot \Pr[\mathcal{M}(X') \in S] + \delta$.

We use the geometric mechanism [14], which is based on the two-sided geometric distribution, as well as the distributed version of this mechanism [17] which is based on the symmetric generalized discrete laplacian distribution [34].

Definition 4 (Probability Distributions).

- *Let $0 < p < 1$ be a real number. We say an integer value random variable Y follows a* ***two-sided geometric distribution*** *denoted as* $\mathsf{Geo}(p)$ *if for any $k \in \mathbb{Z}$*

$$\Pr[Y = k] = \frac{1-p}{1+p} \cdot p^{|k|}.$$

- *Let $0 < p < 1$ and $\beta > 0$ be real numbers. We say an integer value random variable Y follows a* ***symmetric generalized discrete laplacian distribution*** *denoted as* $\mathsf{SGDL}(\beta, p)$ *if for any $k \in \mathbb{Z}$*

$$\Pr[Y = k] = (1-p)^{2\beta} \cdot \sum_{i=|k|}^{\infty} \frac{\Gamma(\beta + i - 1)}{i! \cdot \Gamma(\beta - 1)} \cdot \frac{\Gamma(\beta + i - |k| - 1)}{(i - |k|)! \cdot \Gamma(\beta - 1)} \cdot p^{2i-|k|},$$

  where $\Gamma(\cdot)$ denotes the Gamma function.
- *Let $0 < p < 1$ and $\beta > 0$ be real numbers. We say a non-negative integer value random variable Y follows a* ***negative binomial distribution*** *denoted as* $\mathsf{NB}(\beta, p)$ *if for any $k \in \mathbb{Z}$*

$$\Pr[Y = k] = \frac{\Gamma(\beta + k)}{k! \cdot \Gamma(\beta)} \cdot (1-p)^k \cdot p^{\beta},$$

  where $\Gamma(\cdot)$ denotes the Gamma function.

Lemma 1 (Properties of SGDL). *Let $X \sim Y$ denote equality in distribution between the random variables X, Y*

- ***Case of $\beta = 1$ [34]:*** *For any real number $0 < p < 1$ we have that* $\mathsf{SGDL}(1, p) \sim \mathsf{Geo}(p)$.
- ***Composition of* SGDL *[17]:*** *For any real numbers $0 < p < 1$ and $\beta_1, \beta_2 > 0$ we have that* $\mathsf{SGDL}(\beta_1, p) + \mathsf{SGDL}(\beta_2, p) \sim \mathsf{SGDL}(\beta_1 + \beta_2, p)$.
- ***Sampling of* SGDL *[34]:*** *For any real numbers $0 < p < 1$ and $\beta > 0$ we have that* $\mathsf{SGDL}(\beta, p) \sim \mathsf{NB}(\beta, p) - \mathsf{NB}(\beta, p)$.

The final building block needed to define the geometric mechanism is the concept of *sensitivity of a query*, which for any given metric d is the maximum distance between the evaluation of the query on neighbouring datasets.

Definition 5 (Sensitivity). *Let d be a metric and q a query. The* d-sensitivity *of query q is $\Delta_q := \max_{X, X' \ neighbouring} d(q(X), q(X'))$.*

Then, the geometric mechanism, which is proven to be optimal for counting-like queries [14], is defined as follows.

Lemma 2 (Geometric Mechanism, [14]). *Let $\varepsilon > 0$ be a real number and q a query with sensitivity Δ_q. Then the* geometric mechanism *defined as follows $\mathcal{M}_q(X) := q(X) + e$, with $e \leftarrow \mathsf{Geo}(\exp(-\varepsilon/\Delta_q))$, satisfies $(\varepsilon, 0)$-differential privacy.*

B Security Definitions and Proofs

Secret Sharing. Secret sharing schemes have two main properties: *correctness* and *secrecy*, which are formally defined as follows.

Definition 6 (Correctness and Secrecy of Secret Sharing). *Let* $\mathsf{SS} = (\mathsf{SS.Share}, \mathsf{SS.Combine})$ *be a n-out-of-n secret sharing scheme.*

- *We say it is* correct *if for any element x*

$$\Pr\left[\mathsf{SS.Combine}\left(\left\{[\![x]\!]^{(i)}\right\}_{i\in[n]}\right) \neq x\right] = \mathsf{negl}(\kappa),$$

  where $(\{[\![x]\!]^{(i)}\}_{i\in[n]}) \leftarrow \mathsf{SS.Share}(x)$.
- *We say it is* secret *if for any elements x_0, x_1 and any set of players $\mathcal{S}$ such that $|\mathcal{S}| < n$ the following distributions are indistinguishable $\left\{[\![x_0]\!]^{(i)}\right\}_{i\in\mathcal{S}} \approx \left\{[\![x_1]\!]^{(i)}\right\}_{i\in\mathcal{S}}$ where $(\{[\![x_0]\!]^{(i)}\}_{i\in[n]}) \leftarrow \mathsf{SS.Share}(x_0)$ and $(\{[\![x_1]\!]^{(i)}\}_{i\in[n]}) \leftarrow \mathsf{SS.Share}(x_1)$.*

C Synthetic Database Generation

The datasets upon which our constructions were benchmarked over, are synthetically generated based on real email traffic data from a large corporate group. Since our analysis is more centered on feasibility and overall cost, rather than being interested in absolute fidelity to the real-life scenario, some parameters in the datasets are sampled uniformly at random, to be closer to the average-case scenario. More in details, the datasets are generated as follows.

Private database. We assume the corporate group counts 100 entities gathering 130,000 individuals over 3,000 different locations. Among the 100 entities, at most 3 of them are considered large-size, between 8 and 15 of them are medium-size with a number of employees between 2000 and 5000, and 72 to 91 entities are small-size with a number of employees between 80 and 120. The employees are given a job title picked among 130 possibilities. Associated parameters are:

- `Employee_ID` identifying the individual with an integer in [130000].
- `Location_ID` defining the location with an integer in [3000].
- `Job_Title` defining the job title with an integer in [130].
- `Entity_ID` identifying the entity of the corporate group as an integer in [100]

Additionaly, the following boolean parameters help detect the presence of the employee on site:

- `Work_Saturday`, set to 1 with probability 15%.
- `Work_Sunday`, set to 1 with probability 5% probability.
- `Remote_Work`, set to 1 with probability 65%.

Mail log. To generate the mail log we iterate over every day of the given time frame and sample how many mails each employee would have sent that day from a probability distribution. This distribution is empirically chosen modeling data from a large corporate group following a rounded lognormal distribution with the specific parameters $\mu = \log(9/\sqrt{5})$ and $\sigma = \sqrt{\log(1 + 2/9)}$ for a mean close to 3 mails per day and employee.

Once the number of mails the employee sends in a given day is sampled, some metadata of each mail is simulated as follows:

- `Email_ID`, indexing the mails, assigned incrementally.
- `Sender_ID`, the `Employee_ID` used to generate a new `Email_ID`.
- `Timestamp`, a date and time when the generated email was sent.
- `Sender_Entity`, the `Entity_ID` associated to the `Employee_ID` corresponding to the `Sender_ID`.
- `Receiver_Entity`, with 60% probability of it being the same as `Sender_Entity`, otherwise picked uniformly at random in the set of other entities.
- `Receiver_ID`, picked uniformly at random in the set of employee `Receiver_Entity`.
- `Email_Size`, sampled from a normal distribution with mean $\mu = 102.4$ KB with a standard deviation $\sigma = 24.615$ KB, truncated below 200 B and above 25 MB.