Private Stream Aggregation Revisited

Filipp Valovich, Francesco Aldà
Horst Görtz Institute for IT Security
Faculty of Mathematics
Ruhr-Universität Bochum, Universitätsstrasse 150, 44801 Bochum, Germany
Email: {filipp.valovich, francesco.alda}@rub.de

Abstract

In this work, we investigate the problem of private statistical analysis in the distributed and semi-honest setting. In particular, we study properties of Private Stream Aggregation schemes, first introduced by Shi et al. [27]. These are computationally secure protocols for the aggregation of data in a network and have a very small communication cost. We show that such schemes can be built upon any key-homomorphic weak pseudo-random function. Thus, in contrast to the aforementioned work, our security definition can be achieved in the standard model. In addition, we give a computationally efficient instantiation of this protocol based on the Decisional Diffie-Hellman problem. Moreover, we show that every mechanism which preserves $\epsilon, \delta$-differential privacy provides computational $\epsilon, \delta$-differential privacy when it is executed through a Private Stream Aggregation scheme. Finally, we introduce a novel perturbation mechanism based on the Skellam distribution that is suited for the distributed setting, and compare its performances with those of previous solutions.

1 Introduction

The framework of statistical disclosure control aims at providing strong privacy guarantees for the records stored in a database while enabling accurate statistical analyses to be performed. In recent years, differential privacy has become one of the most important paradigms for privacy-preserving statistical analyses. Generally, the notion of differential privacy is considered in the centralised setting where we assume the existence of a trusted curator [4, 8, 10, 21] who collects data in the clear, perturbs it properly (e.g. by adding Laplace noise) and publishes it. In this way, the output statistics are not significantly influenced by the presence (resp. absence) of a particular record in the database.

In this work, we study how to preserve differential privacy when we cannot rely on a trusted curator. In this so-called distributed setting, the users have to send their data to an untrusted aggregator. Preserving differential privacy and achieving high accuracy in the distributed setting is of course harder than in the centralised setting, since the users have to execute a perturbation mechanism on their own. In order to achieve the same

*The research was supported by the DFG Research Training Group GRK 1817/1
accuracy as provided by well-known techniques in the centralised setting, the work by Shi et al. [27] introduces the Private Stream Aggregation (PSA) scheme, a cryptographic protocol which enables each user to securely send encrypted time-series data to an aggregator. The aggregator is then able to decrypt the aggregate of all data in each time step, but cannot retrieve any further information about the individual data. Using such a protocol, the task of perturbation can be split among the users, such that differential privacy is preserved and high accuracy is guaranteed. For a survey of applications of this protocol, we refer to [27].

In [27], a PSA scheme for sum queries is provided and some strong security guarantees under the Decisional Diffie-Hellman assumption are shown. However, this instantiation has some limitations. First, the security only holds in the random oracle model; second, its decryption algorithm requires the solution of the discrete logarithm in a given range, which can be very time-consuming if the number of users and the plaintext space are large. Moreover, since a PSA scheme provides computational security, the perturbation mechanism in use can only provide a computational version of differential privacy, a notion first introduced by Mironov et al. [22]. In [27], however, a connection between the security of a PSA scheme and differential privacy is not explicitly shown. In a subsequent work by Chan et al. [7], this connection is still not completely established, since the polynomial-time reduction between an attacker against a PSA scheme and a database distinguisher is missing.

Objectives. In order to overcome the limitations of the construction in [27], in this work we address the following problems.

- We want to give sufficient conditions for a PSA scheme which has a security guarantee in the standard model.
- According to these conditions, we want to construct a concrete instantiation consisting of efficient algorithms, even when the number of users and the plaintext space become large.
- We aim at showing that an information-theoretical differentially private mechanism preserves computational differential privacy when it is executed through a (computationally) secure PSA scheme.
- We want to investigate differentially private mechanisms suitable for an execution through a PSA scheme.

Contributions. We achieve the aforementioned goals in the following manner. In order to derive sufficient conditions for PSA schemes with a certain security guarantee, we lower the requirements of Aggregator Obliviousness from [27] by abrogating the attacker’s possibility to adaptively compromise users during the execution of a PSA scheme with time-series data. We show that a PSA scheme for achieving this lower security level can be built upon any key-homomorphic weak pseudo-random function. Since weak pseudo-randomness can be achieved in the standard model, this condition also enables secure schemes in the standard model. In particular, we can build a key-homomorphic weak pseudo-random function based on the Decisional Diffie-Hellman assumption in the group of quadratic residues modulo a squared safe prime. This function is used for the construction of a PSA scheme for sum queries. By comparing the running times
and practical performances of our PSA scheme and the one given by Shi et al. [27] at the same security level, we find that our solution provides a significant speed-up for decryption when the plaintext space is large while decelerating the encryption only by a constant factor.

Reduction-based security proofs for cryptographic schemes usually require an attacker in the corresponding security game to send two different plaintexts (or plaintext collections) to a challenger. The adversary receives then back a ciphertext which is the encryption of one of these collections and has to guess which one it is. In any security definition for a PSA scheme, these collections must satisfy a particular requirement (i.e. they must lead to the same aggregate), since the attacker has the capability to decrypt them (different aggregates would make the adversary’s task trivial). In general, however, this requirement cannot be satisfied in the context of differential privacy. Introducing a novel kind of security reduction which deploys a biased coin flip, we can show that, whenever a randomised perturbation procedure is involved in a PSA scheme, the requirement of having collections with equal aggregate can be abolished. This result can be generalised to any cryptographic scheme with such a requirement. Using this property, we are able to show that if a mechanism preserves differential privacy, then it preserves computational differential privacy when it is used as a randomised perturbation procedure in a PSA scheme.

Finally, we compare three mechanisms: the Geometric mechanism from [27], the Binomial mechanism from [9] and the Skellam mechanism introduced in this work. All three mechanisms preserve differential privacy and make use of discrete probability distributions. Therefore, they are well-suited for an execution through a PSA scheme. For generating the right amount of noise among all users, these mechanisms apply two different approaches. While in the Geometric mechanism, with high probability, only one user generates the noise necessary for differential privacy, the Binomial and Skellam mechanisms allow all users to generate noise of small variance, that sums up to the required value for privacy. We show that for high privacy levels, the theoretical error bound of the Skellam mechanism is slightly better than that of the other two. At the same time, we provide experimental results showing that the Geometric and Skellam mechanisms have a comparable accuracy in practice, while beating the one of the Binomial mechanism.

Related Work. As pointed out above, our contributions are mostly related to the work of Shi et al. [27] and Chan et al. [7]. Privacy-preserving aggregation of time-series data in the presence of an untrusted aggregator has also been studied in various other works, e.g. [3, 11, 18, 19, 16]. Beimel et al. [3] and Eigner et al. [11] show that secure multi-party computation techniques can be used for data aggregation under differential privacy. These techniques usually have a high communication cost, whereas PSA requires each user to send exactly one message per time-step. The protocol given by Rastogi et al. [25] is based on the threshold Paillier cryptosystem. It requires an extra round of interaction between the users and the aggregator in every time-step in order to decrypt the sum queries. In contrast, PSA requires the users to interact with the aggregator only for sending the ciphertexts. Ács et al. [2] use an additive homomorphic encryption scheme for sending time-series data, but it requires the generation of a pair of encryption/decryption keys for each pair of users. Moreover, reuse of key pairs for different time-steps potentially leads to security breaches. In a PSA scheme each user gets only one encryption key, which can be securely used for every time-step. Using
additive homomorphic encryption, Rieffel et al. [26] construct a scheme which does not require extra rounds of interaction, but is not fully resistant against collusions and the cost of computation and storage is roughly equal to the number of compromised users that is tolerable by the system. Li et al. [18, 19] use the homomorphic encryption scheme given by Castelluccia et al. [5, 6] in order to construct an efficient protocol for sending data in mobile sensing applications. This scheme is resistant against collusions, but each user has to store multiple keys, depending on the number of compromised users in the network. Moreover, for encryption and decryption the scheme requires the computation of as many pseudo-random values as the number of keys in the network, making the computational effort for the analyst rather high. Thus, the costs of this scheme depend on the number of compromised users. A PSA scheme is fully resistant against any number of collusions and furthermore, we provide a solution, where the computation and storage costs are independent of the number of users. Joye et al. [16] provide a protocol with the same security guarantees as in [27] in the random oracle model. The security of their scheme relies on the DCR assumption (rather than DDH as in [27]) and as a result, in the security reduction they can remove a factor which is cubic in the number of users. However, their scheme involves a trusted party for setting some public parameters. In this work we provide an instantiation of our generic PSA construction, which is similar to the one in [16] but relies on the DDH assumption. While in our generic security reduction we cannot avoid the cubic factor in the number of users, our construction does not involve any trusted party and has security guarantees in the standard model.

Another series of works deals with a distributed generation of noise for preserving differential privacy. Dwork et al. [9] consider the Gaussian distribution for splitting the task of noise generation among all users. Their proposed scheme requires more interactions between the users than our solution. In [2], privacy-preserving data aggregation is applied to smart metering and the generation of Laplace noise is performed in a distributed manner, since each meter simply generates the difference of two Gamma distributed random variables as a share of a Laplace distributed random variable. In [25], each user generates a share of Laplace noise by generating a vector of four Gaussian random variables. For a survey of the mechanisms given in [25] and [2], we refer to [14]. However, the aforementioned mechanisms generate noise drawn according to continuous distributions, but for the use in a PSA scheme, discrete noise is required. Therefore, we consider proper discrete distributions and compare their performances for private statistical analyses.

2 Preliminaries

2.1 Problem statement

In this work, we consider a distributed and semi-honest setting where \( n \) users are asked to participate in some statistical analyses but do not trust the data analyst (or aggregator), who is assumed to be honest but curious. Therefore, the users cannot provide their own data in the clear. Moreover, they communicate solely and independently with the untrusted aggregator, who wants to analyse the users data by means of queries in time-series and aims at obtaining answers as accurate as possible. More specifically, assume that the data items belong to a data universe \( D \). For a sequence of time-steps
In connection with a perturbation mechanism, this scheme ensures that the analyst is only able to learn a noisy aggregate of the users’ data (as close as possible to the real answer $f(D)$) and nothing else. In contrast to common secure multi-party techniques \cite{12, 13, 20}, this protocol requires each user to send to the analyst only one message per query.

### 2.2 Definitions

We consider a database as an element $D \in \mathcal{D}^n$, where $\mathcal{D}$ is the data universe and $n$ is the number of users. Since $D$ may contain sensitive information, the users want to protect their privacy. Therefore, a privacy-preserving mechanism must be applied. Unless stated differently, we always assume that a mechanism is applied in the distributed setting. Differential privacy \cite{10} is a well-established notion for privacy-preserving statistical analyses. We recall that a randomised mechanism preserves differential privacy if its application on two adjacent databases, i.e. databases which differ in one entry only, leads to close distributions of the output.

**Definition 1** (Differential Privacy \cite{10}). Let $\mathcal{R}$ be a (possibly infinite) set and let $n \in \mathbb{N}$. A randomised mechanism $A : \mathcal{D}^n \rightarrow \mathcal{R}$ preserves $(\epsilon, \delta)$-differential privacy (short: DP), if for all adjacent databases $D_0, D_1 \in \mathcal{D}^n$ and all $R \subseteq \mathcal{R}$:

$$\Pr[A(D_0) \in R] \leq e^\epsilon \cdot \Pr[A(D_1) \in R] + \delta.$$ 

The probability space is defined over the randomness of $A$.

The additional parameter $\delta$ is necessary for mechanisms which cannot preserve $\epsilon$-DP (i.e. $(\epsilon, 0)$-DP) for certain cases. However, if the probability that these cases occur is bounded by $\delta$, then the mechanism preserves $(\epsilon, \delta)$-DP.

In the literature, there are well-established mechanisms for preserving differential privacy, e.g. the Laplace mechanism \cite{10} and the Exponential mechanism \cite{21}. In order to privately evaluate a query, these mechanisms draw noisy values according to some distribution depending on the query’s global sensitivity.

**Definition 2** (Global Sensitivity \cite{10}). The global sensitivity $S(f)$ of a query $f : \mathcal{D}^n \rightarrow \mathcal{O}$ for a finite or countably infinite set of possible outputs (i.e. answers to the query) $\mathcal{O}$.
\( f : \mathcal{D}^n \rightarrow \mathbb{R}^k \) is defined as
\[
S(f) = \max_{D_0, D_1 \text{ adjacent}} \| f(D_0) - f(D_1) \|_1.
\]

In particular, we will consider sum queries \( f_D : \mathcal{D}^n \rightarrow \mathbb{Z} \) defined as \( f_D(D) := \sum_{i=1}^n d_i \), for \( D = (d_1, \ldots, d_n) \in \mathcal{D}^n \) and \( \mathcal{D} \subseteq \mathbb{Z} \).

For measuring how well the output of a mechanism estimates the real data with respect to a particular query, we use the notion of \((\alpha, \beta)\)-accuracy.

**Definition 3** (Accuracy [29]). The output of a mechanism \( A \) achieves \((\alpha, \beta)\)-accuracy for a query \( f : \mathcal{D}^n \rightarrow \mathbb{R} \) if for all \( D \in \mathcal{D}^n \):
\[
\Pr[|A(D) - f(D)| \leq \alpha] \geq 1 - \beta.
\]
The probability space is defined over the randomness of \( A \).

The use of a cryptographic protocol for transferring data provides a computational security level. If such a protocol is applied for preserving differential privacy, this implies that only a computational level of differential privacy can be provided. Our definition of computational differential privacy follows the notion of Chan et al. [7].

**Definition 4** (Computational Differential Privacy [7]). Let \( \kappa \) be a security parameter and \( n \in \mathbb{N} \) with \( n = \text{poly}(\kappa) \). A randomised mechanism \( A : \mathcal{D}^n \rightarrow \mathbb{R} \) preserves computational \((\epsilon, \delta)\)-differential privacy (short: CDP), if for all adjacent databases \( D_0, D_1 \in \mathcal{D}^n \) and all probabilistic polynomial-time distinguishers \( D_{\text{CDP}} \):
\[
\Pr[D_{\text{CDP}}(1^\kappa, A(D_0)) = 1] \leq \epsilon \cdot \Pr[D_{\text{CDP}}(1^\kappa, A(D_1)) = 1] + \delta + \text{neg}(\kappa),
\]
where \( \text{neg}(\kappa) \) is a negligible function in \( \kappa \). The probability space is defined over the randomness of \( A \) and \( D_{\text{CDP}} \).

The notion of computational differential privacy is a natural computational-indistinguishability-extension of the information-theoretical definition. The advantage is that preserving differential privacy only against bounded attackers helps to substantially reduce the error of the answer provided by the mechanism. In Section 3, we investigate how to obtain a computationally secure protocol which allows the analyst to compute only the aggregate of all users’ data and no further information. The scheme for sum queries we are going to construct uses a special mapping into a group, which we define formally.

**Definition 5** (\( v \)-isomorphic embedding). An injective mapping \( \varphi : \{-v, \ldots, v\} \rightarrow V \), where \( (V, \circ) \) is a group, is a \( v \)-isomorphic embedding if for all \( n \in \mathbb{N} \) and all finite sequences \( (a_i)_{i=1}^n \) of values in \( \{-v, \ldots, v\} \) with \( |\sum_i a_i| \leq v \):
\[
\varphi \left( \sum_{i=1}^n a_i \right) = \varphi(a_1) \circ \ldots \circ \varphi(a_n).
\]

From this definition it is clear that a \( v \)-isomorphic embedding is also \( v' \)-isomorphic for every integer \( 0 < v' \leq v \). In the analysis of the secure protocol, we furthermore make use of the following definition.
Definition 6 (Weak PRF [24]). Let $\kappa$ be a security parameter. Let $A, B, C$ be sets. A family of functions

$$\mathcal{F} = \{F_a | F_a : B \to C \}_{a \in A}$$

is called a weak pseudo-random function (PRF) family if for all probabilistic polynomial-time algorithms $D_O$ with oracle access to $O(\cdot)$ (where $O(\cdot) \in \{F_a(\cdot), \text{rand}(\cdot)\}$) on any polynomial number of uniformly chosen inputs, we have:

$$|\Pr[D_{PRF}^{F_a}(\kappa) = 1] - \Pr[D_{PRF}^{\text{rand}}(\kappa) = 1]| \leq \text{neg}(\kappa),$$

where $a \in R$ and $\text{rand} \in R \{f | f : B \to C\}$ is a random mapping from $B$ to $C$.

2.3 Mechanism overview

In this work we prove the following result by showing the connection between a key-homomorphic weak pseudo-random function and a differentially private mechanism for sum queries.

Theorem 1. Let $\epsilon > 0$, $w < w' \in \mathbb{Z}$, $m, n \in \mathbb{N}$ with $\max\{|w|, |w'|\} < m$. Let $D = \{w, \ldots, w'\}$ and $f_D$ be a sum query. If there exist groups $G' \subseteq G$, a key-homomorphically weak pseudo-random function family mapping into $G'$ and an efficiently computable and efficiently invertible $mn$-isomorphic embedding

$$\varphi : \{-mn, \ldots, mn\} \to G,$$

then there exists an efficient mechanism for $f_D$ that preserves $(\epsilon, \delta)$-CDP for any constant $0 < \delta < 1$ with an error bound of $O(S(f_D)/\epsilon)$ and requires each user to send exactly one message.

As already described, we want the untrusted analyst to be able to learn some aggregated statistics $f_D(D)$ but no additional information about each user’s data. Assume that we can design a cryptographic protocol that achieves the aforementioned goal. If we furthermore aim at preserving $(\epsilon, \delta)$-DP, it would be sufficient to add a single copy of (properly distributed) noise $Y$ to the value $f_D(D)$. Since we cannot add such noise once the aggregate has been computed, the users have to generate and add noise to their original data in such a way that the sum of the perturbations has the same distribution as $Y$. For this purpose, we see two different approaches. In the first approach, there is a small probability (depending on the fixed parameter $\gamma$) for each user to add noise sufficient to preserve the privacy of the entire statistics. This probability is calibrated in such a way only one of the $n$ users is expected to add noise at all. Shi et al. [27] investigate this method using the Geometric mechanism. In the second approach, each user generates noise of small variance (again depending on $\gamma$), such that the sum of all noisy terms has enough variance for preserving differential privacy. For this aim, we need discrete probability distributions which are closed under convolution and are known to provide differential privacy. The Binomial mechanism [9] and the Skellam mechanism introduced in this work serve these purposes. In both approaches, the error which is introduced is reasonably small and similar theoretical bounds can be provided. For details, see Section [5].
For a particular time-step, let the users’ values be of the form \( x_i = d_i + r_i \), where \( d_i \in \mathcal{D} \) is the original data of the user \( i \) and \( r_i \) is her noisy value. In the privacy analysis, it is reasonable to assume that \( r_i = 0 \) for the \((1-\gamma)\cdot n\) compromised users, since this can only increase their chances to infer some information about the non-compromised users. In order to send the values to the data analyst, the users perform a PSA scheme. First, each user encrypts her own time-series data and sends the ciphertexts to the data analyst. After a distributed key exchange, the evaluation of a single query (i.e. a query analysed in one time-step) requires each user to send exactly one message. The data analyst appropriately aggregates the ciphertexts of all users for a particular time-step and then decrypts the sum of the users’ values \( \sum_{i=1}^{n} x_i \). From the ciphertexts, the data analyst is not able to learn any additional information about the values of the users, except for the auxiliary information obtained from the compromised users. In this way, there is no privacy-breach if only one user adds the entirely needed noise (first approach) or if the non-compromised users generate noise of low variance (second approach), since the single values are encrypted and the analyst cannot learn anything about them, except for their aggregate. Due to the use of a cryptographic protocol, the plaintexts have to be discrete. This is the reason why we use discrete distributions for generating the noisy values \( r_i \).

The perturbation of data potentially yields larger values \( x_i \) due to the (possibly) infinite domain of the underlying probability distribution. Depending on the variance, we therefore need to choose a sufficiently large interval \( \mathcal{D} = \{-m, \ldots, m\} \) as plaintext space, where \( m > \max\{|w|, |w'|\} \) such that \( |x_i| \leq m \) for all \( i = 1, \ldots, n \) with high probability. In the following, we always assume that \( \mathcal{D} \) is a subinterval of \( \mathcal{D} \).

Since the protocol used for the data transmission is computationally secure, the entire mechanism preserves \((\epsilon, \delta)\)-CDP.

3 Private Stream Aggregation

In this section, we define the Private Stream Aggregation scheme and give a security definition for it. Thereby, we mostly follow the concepts introduced by Shi et al. [27], though we deviate in a few points. Afterwards, we give a condition for the existence of secure PSA schemes. Moreover, we give a concrete and efficient instantiation of a secure PSA scheme in the standard model.

3.1 The definition of Private Stream Aggregation and its security

Private Stream Aggregation. A PSA scheme is a protocol for safe distributed time-series data transfer which enables the receiver to learn only the aggregate \( f(\mathcal{D}) \) of a query \( f : \mathcal{D}^n \rightarrow \mathcal{O} \) over some distributed (and possibly perturbed) database \( \mathcal{D} \in \mathcal{D}^n \).

Such a scheme needs a key exchange protocol for all \( n \) users together with the analyst as a precomputation, and requires each user to send exactly one message per query. For the definition of PSA, we follow [27].

Definition 7 (Private Stream Aggregation). Let \( \kappa \) be a security parameter and \( n \in \mathbb{N} \) with \( n = \text{poly}(\kappa) \). A Private Stream Aggregation scheme \( \Sigma = (\text{Setup}, \text{PSAEnc}, \text{PSADec}) \) is defined by three probabilistic polynomial-time Algorithms:
**Setup:** $(pp, T, s, s_1, \ldots, s_n) \leftarrow \text{Setup}(\kappa)$, where $pp$ are public parameters of the system, $T$ is a set of time-steps and $s, s_1, \ldots, s_n$ are private keys.

**PSAEnc:** For time-step $t \in T$ and all $i = 1, \ldots, n$:
\[
c_{i,t} \leftarrow \text{PSAEnc}_{s_i}(t, x_i) \text{ for a data value } x_i \in \hat{D}.
\]

**PSADec:** For time-step $t \in T$, ciphertexts $c_{1,t}, \ldots, c_{n,t} \in C$, where $C$ is the range of PSAEnc, and a query $f : \hat{D}^n \rightarrow O$ compute
\[
f(x'_1, \ldots, x'_n) = \text{PSADec}_s(t, c_{1,t}, \ldots, c_{n,t}).
\]
For all $t \in T$ and $x_1, \ldots, x_n \in \hat{D}$:
\[
\text{PSADec}_s(t, \text{PSAEnc}_{s_1}(t, x_1), \ldots, \text{PSAEnc}_{s_n}(t, x_n)) = f(x_1, \ldots, x_n).
\]

The Setup-phase has to be carried out just once and for all, and can be performed with a secure multi-party protocol among all users and the analyst. In all other phases, no communication between the users is needed. The system parameters $pp$ are public and constant for all time-steps with the implicit understanding that they are used in $\Sigma$. Every user encrypts her data $x_i$ with her own secret key $s_i$ and sends the ciphertext to the analyst. If the analyst receives the ciphertexts of all users for a time-step $t$, it can compute the aggregate, i.e. the evaluation of the query $f$, of the users' data with the decryption key $s$.

**Security.** Since our model allows the analyst to compromise users, the aggregator can obtain auxiliary information about the data of the compromised users or their secret keys. Even then a secure PSA scheme should release no more information than the aggregate of the non-compromised users’ data in a single time-step. Informally, a PSA scheme $\Sigma$ is secure if every probabilistic polynomial-time algorithm, with knowledge of the analyst’s and compromised users’ keys and with adaptive encryption queries, has only negligible advantage in distinguishing between the encryptions of two databases $\hat{D}_0, \hat{D}_1$ of its choice, where $f(\hat{D}_0) = f(\hat{D}_1)$. We can assume that an adversary knows the secret keys of the entire compromised coalition. If the protocol is secure against such an attacker, then it is also secure against an attacker without the knowledge of every key from the coalition. Thus, in our security definition we consider the most powerful adversary. In what follows, let $f_{|X} : \hat{D}^{|X|} \rightarrow O$ denote the evaluation of a query $f : \hat{D}^n \rightarrow O$ with input $\hat{D} \in \hat{D}^n$ over a subset $X \subseteq [n]$ of users. The security definition is similar to the one in [27].

**Definition 8 (Security of PSA).** Let $T$ be a probabilistic polynomial-time adversary for a PSA scheme $\Sigma = (\text{Setup}, \text{PSAEnc}, \text{PSADec})$ and let $f : \hat{D}^n \rightarrow O$ be a statistical query over the set $\hat{D}$. Let $T$ be the set of time-steps for possible data analyses. We define the following security game between a challenger and the adversary $T$.

**Setup.** The challenger runs the Setup algorithm on input security parameter $\kappa$ and returns public parameters $pp$, time-steps $T$ with $|T| = \text{poly}(\kappa)$ and secret keys $s, s_1, \ldots, s_n$. It sends $\kappa, pp, T, s$ to $T$. 

**Queries.** The challenger flips a random bit \( b \in \{0, 1\} \). \( \mathcal{T} \) chooses \( U \subseteq [n] \) and sends it to the challenger which returns \((s_i)_{i \in [n] \setminus U}\). \( \mathcal{T} \) is allowed to query \((i, t, x_i)\) with \( i \in U, t \in T, x_i \in \hat{D} \) and the challenger returns

\[
\text{PSAEnc}_{s_i}(t, x_i).
\]

**Challenge.** \( \mathcal{T} \) chooses \( t^* \in T \) such that no encryption query at \( t^* \) was made. (If there is no such \( t^* \) then the challenger simply aborts.) \( \mathcal{T} \) queries two different tuples \((x_i^{[0]}), (x_i^{[1]})_{i \in U}\) with

\[
f_U((x_i^{[0]}))_{i \in U}) = f_U((x_i^{[1]}))_{i \in U}).
\]

For all \( i \in U \) the challenger returns

\[
c_i, t^* \leftarrow \text{PSAEnc}_{s_i}(t^*, x_i^{[b]}).
\]

**Queries.** \( \mathcal{T} \) is allowed to make the same type of queries as before with the restriction that no encryption query at \( t^* \) can be made.

**Guess.** \( \mathcal{T} \) outputs a guess about \( b \).

The adversary wins the game if it correctly guesses \( b \). A PSA scheme is secure if no probabilistic polynomial-time adversary \( \mathcal{T} \) has more than negligible advantage (with respect to the parameter \( \kappa \)) in winning the above game.

Encryption queries are made only for \( i \in U \), since knowing the secret key for all \( i \in [n] \setminus U \) the adversary can encrypt a value autonomously. If encryption queries in time-step \( t^* \) were allowed, then no deterministic scheme would be secure. The adversary \( \mathcal{T} \) can determine the original data of all \( i \in [n] \setminus U \) for every time-step, since it knows \((s_i)_{i \in [n] \setminus U}\). Then \( \mathcal{T} \) can compute the aggregate of the non-compromised users’ data. For example, when \( f = f_B \) is a sum query we have

\[
f_U((\hat{D})) = \text{PSADec}_{s_i}(t, c_{1,t}, \ldots, c_{n,t}) - f_U((\hat{D}))
\]

where \( \hat{D} = (x_1, \ldots, x_n) \) and \( c_i,t \) is the encryption of \( x_i \) for all \( i \in [n] \). On the other hand, if there is a user’s ciphertext which \( \mathcal{T} \) does not receive, then it cannot obtain the aggregate for the correspondent time-step.

The security definition indicates that \( \mathcal{T} \) cannot distinguish between the encryptions of two different data collections \((x_i^{[0]}), (x_i^{[1]})_{i \in U}\) with the same aggregate at time-step \( t^* \). For proving that a secure PSA scheme can be used for computing differentially private statistics with small error, we have to slightly modify the security game such that an adversary may choose adjacent (and non-perturbed) databases, as it is required in the definition of differential privacy. For details, see Section 4.

Definition 8 differs from the definition of Aggregator Obliviousness [27] since we require the adversary to specify the set \( U \) of non-compromised users before making any query, i.e. we do not allow the adversary to determine \( U \) adaptively.

### 3.2 Feasibility of efficient and secure Private Stream Aggregation

Using a secure PSA scheme, we ensure that the transmitted data of non-compromised users do not disclose sensitive information other than their aggregate. We now state the condition for the existence of secure (as in Definition 5) PSA schemes for sum queries.
Theorem 2. Let $\kappa$ be a security parameter, and $m, n \in \mathbb{N}$ with $\log(m) = \text{poly}(\kappa), n = \text{poly}(\kappa)$. Let $(G, \cdot), (S, *)$ be finite groups and $G' \subseteq G$. For some finite set $M$, let

$$\mathcal{F} = \{F_s | F_s : M \to G'\}_{s \in S}$$

be a family of functions which are homomorphic over $S$ and

$$\varphi : \{-mn, \ldots, mn\} \to G$$

an $mn$-isomorphic embedding. If $\mathcal{F}$ is a weak PRF family, then the following PSA scheme $\Sigma = \text{Setup}, \text{PSAEnc}, \text{PSADec}$ is secure:

**Setup:** $(pp, T, s, s_1, \ldots, s_n) \leftarrow \text{Setup}(1^\kappa)$, where $pp$ are parameters of $G, G', S, M, \mathcal{F}, \varphi$.

The keys are $s_i \in_R S$ for all $i \in [n]$ with $s = (s_1^n s_1)^{-1}$ and $T \subseteq M$ such that all $t \in T$ are chosen uniformly at random from $M$.

**PSAEnc:** Compute $c_{i,t} = F_{s_i}(t) \cdot \varphi(x_i)$ in $G$, where $x_i \in \mathbb{D} = \{-m, \ldots, m\}, t \in T$.

**PSADec:** Compute $\varphi(\sum_{i=1}^n x_i) = F_s(t) \cdot c_{1,t} \cdot \ldots \cdot c_{n,t}$ and invert.

Thus, we need a key-homomorphic weak PRF and a mapping which homomorphically aggregates all users’ data. Since every data value is at most $m$, the scheme correctly retrieves the aggregate, which is at most $m \cdot n$, by the $mn$-isomorphic property of $\varphi$. Importantly, the product of all pseudo-random values $F_s(t), F_{s_1}(t), \ldots, F_{s_n}(t)$ is the neutral element in the group $G$ for all $t \in T$. Since the values in $T$ are uniformly distributed in $M$, it is enough to require that $\mathcal{F}$ is a weak PRF family. Thus, the statement of this theorem does not require a random oracle.

The proof of Theorem 2 works with a sequence of games and builds on the ideas of [27]. The details of the proof are given in Appendix A. Here we just give the main ideas. In the first step (Lemma 9) we construct a real-or-random version of the PSA security game, where encryption is performed equally likely with a weak PRF or a random function. We show that winning the PSA security game is at least as hard as winning its real-or-random version. The second step (Lemma 10) shows that the plaintext dependence of the ciphertexts generated in the game can be abolished. Since we are dealing with a non-adaptive security definition there is no need of simulating the random choice of time-steps by programming a random oracle as it is required in the proof of Aggregator Obliviousness by Shi et al. [27]. Therefore, in contrast to [27], our result does not rely on a random oracle and the full proof works in the standard model. In the last step, (Lemma 11) using the hybrid argument, we show that winning the game is at least as hard as distinguishing the weak PRF from a random function. Here the adversary’s specification of $U$ before making the first query allows the PRF distinguisher to be consistent with real random values or pseudo-random values in its replies to the queries. All in all, we get that winning the first game with our construction is at least as hard as distinguishing the weak PRF from a random function, completing the proof of Theorem 2.

### 3.3 An efficient Private Stream Aggregation scheme

We give an instantiation of a secure PSA scheme consisting of efficient algorithms. Its security is based on the Decisional Diffie-Hellman (DDH) problem.
Example 1. Let $q > m \cdot n$ and $p = 2 \cdot q + 1$ be large primes. Let furthermore $G = \mathbb{Z}_{p^2}^*$, $S = \mathbb{Z}_{pq}$, $M = G' = \mathbb{QR}_{p^2}$ and $g \in \mathbb{Z}_{p^2}^*$ with $\operatorname{ord}(g) = pq$. Then $g$ generates the group $M = G' = \mathbb{QR}_{p^2}$ of quadratic residues modulo $p^2$. In this group DDH is assumed to be hard. Then we define

- $\{pp\} = (g,p)$. Choose keys $s_1, \ldots, s_n \in \mathbb{Z}_{pq}$ and $s = -\sum_{i=1}^n s_i \mod pq$. Let $T \subset M$, i.e. $t$ is a power of $g$ for every $t \in T$.
- $F_{s_i}(t) = t^{s_i} \mod p^2$. This is a weak PRF under the DDH assumption, as can be proven using arguments similar to the ones in [23].
- $\varphi(x_i) = 1 + p \cdot x_i \mod p^2$, where $-m \leq x_i \leq m$. (It is easy to see that $\varphi$ is an $mn$-isomorphic embedding.)

For aggregation, we compute $X \in \{1 - p \cdot mn, \ldots, 1 + p \cdot mn\}$ with

$$X \equiv F_s(t) \cdot \prod_{i=1}^n F_{s_i}(t) \cdot \varphi(x_i) \equiv \prod_{i=1}^n (1 + p \cdot x_i)$$

$$\equiv 1 + p \cdot \sum_{i=1}^n x_i + p^2 \cdot \sum_{i,j \in [n], j \neq i} x_i x_j + \ldots + p^n \cdot \prod_{i=1}^n x_i$$

$$\equiv 1 + p \cdot \sum_{i=1}^n x_i \mod p^2$$

and decrypt $\sum_{i=1}^n x_i = \frac{1}{p}(X - 1)$ over the integers.

The difference to the scheme introduced in [27] lies on the map $\varphi$. Whereas the PRF in [27] works similarly (the underlying group $G$ is $\mathbb{Z}_p^*$ rather than $\mathbb{Z}_{p^2}^*$), the aggregational function is defined by

$$\varphi(x_i) = g^{x_i} \mod p,$$

which requires to solve the discrete logarithm modulo $p$ for decrypting. In contrast, our efficient construction only requires a subtraction and a division over the integers.

Remark 1. In the random oracle model, the construction shown in Example 1 achieves the stronger notion of Aggregator Obliviousness, which is the adaptive version of our security definition (for details, see the proof in Appendix A of [27]). The same proof can be applied to our instantiation by simply replacing the map $\varphi$ involved and using a strong version of the PRF $F^1$.

Note that for a given $p$, the running time of the decryption in our scheme does not depend on $m$, so it provides a small running time even if $m$ is exponentially large. At the same time, the decryption of the scheme in [27] can also be performed efficiently, even if $m$ is superpolynomial in the security parameter: discretise the plaintext space into $\sqrt{m}$ equidistant values and let each user choose the value nearest to her original

---

1For showing Aggregator Obliviousness we would have to substitute the choice of random values $t$ by a hash function $H : M \rightarrow \mathbb{QR}_{p^2}$ modeled as a random oracle for some domain $M$. Therefore the PRF would be $F_{s_i}(t) = H(t)^{s_i} \mod p^2$ which is the strong version of the weak PRF in Example 1. In this case, all $t$ may be chosen in a deterministic way.
| Length of $p$ | 1024-bit | 2048-bit | 4096-bit |
|--------------|-----------|-----------|-----------|
| [27]         | 1.1 ms    | 7.5 ms    | 57.0 ms   |
| This work    | 3.9 ms    | 29.4 ms   | 225.0 ms  |

Table 1: Time for encryption

| $m$          | $10^0$ | $10^1$ | $10^2$ | $10^3$ | $10^4$ |
|--------------|--------|--------|--------|--------|--------|
| [27] b.-f.   | 0.04 s | 0.24 s | 2.67 s | 28.97 s| 381.05 s|
| This work    | 0.09 s | 0.08 s | 0.08 s | 0.09 s | 0.08 s |

Table 2: Time for decryption (2048 bit, $n = 1000$)

one as input for encryption. The aggregated value has the correct expectation, but the decryption algorithm has to search only in a range of $n^{1.5}$ values. However, this method causes a superlinear time-dependence on $n$ for the decryption and induces an additional aggregation error due to discretisation.

We compare the practical running times for encryption and decryption of the scheme in [27] with the algorithms of our scheme in Table 1 and Table 2 respectively. Here, let $m$ denote the size of the plaintext space. Encryption is compared at different security levels with $m = 1$. For comparing the decryption time, we fix the security level and the number of users and let $m$ be variable.

All algorithms were executed on an Intel Core i5, 64-bit CPU at 2.67 GHz. We compared the schemes at the same security level, assuming that the DDH problem modulo $p$ is as hard as modulo $p^2$, i.e. we used the same value for $p$ in both schemes. For different bit-lengths of $p$, we observe that the encryption of our scheme is roughly 4 times slower than the encryption in [27]. The running time of our decryption algorithm is widely dominated by the aggregation phase. Therefore it is clear, that it linearly depends on $n$. Using a 2048-bit prime and fixing $n = 1000$, the running time of the decryption in our scheme is less than 0.1 second for varying values of $m$. In contrast, the time for the brute-force decryption in [27] grows roughly linearly in $n$.

As observed in [27], using Pollard’s lambda method would reduce the running time for decryption of [27] to about $\sqrt{mn}$. Nevertheless, our scheme provides a speed-up of $\sqrt{m/n}$ whenever $m$ is larger than $n$, while the encryption is decelerated only by a constant factor.

4 Achieving Computational Differential Privacy

**Notation 1.** Let $\kappa$ be a security parameter. If an expression $\omega = \omega(\kappa)$ is non-negligible in $\kappa$ (i.e. if $\omega > 1/poly(\kappa)$), then we write $\omega \succ neg(\kappa)$.

In this section, we describe how to preserve computational differential privacy using a PSA scheme. As described above, in the work by Chan et al. [2] the polynomial-time reduction between an attacker against the security of a PSA scheme and an attacker against differential privacy is missing. In this section we provide an appropriate reduction. The content of this section is independent of Theorem [2]. Specifically, let $\mathcal{A}$ be a mechanism which, given some event $Good$, evaluates a statistical query $f : D^n \rightarrow O$ over a database $D \in D^n$ preserving $\epsilon$-DP. Furthermore, let $\Sigma$ be a secure PSA scheme for $f$. 
We show that \( A \) executed through \( \Sigma \) preserves \( \epsilon \text{-CDP} \) given \( \text{Good} \). Let \( \text{Bad} = \overline{\text{Good}} \) and assume \( \Pr[\text{Bad}] \leq \delta \). In Section 5 we will give instantiations of such a mechanism \( A \) and show that they preserve \( (\epsilon, \delta) \text{-CDP} \) unconditionally if executed through \( \Sigma \). For simplicity, in this section we focus on sum queries, but our analysis can easily be extended to more general statistical queries. Our technique involves a reduction-based proof using a biased coin toss and is of independent interest.

### 4.1 Redefining the security of Private Stream Aggregation

Let us first modify the security game in Definition 8 in the following way. Let \( \text{game 1} \) be the original game from Definition 8. Let \( p \in (0, 1) \) and \( P = \max\{p, 1 - p\} \). The \( P\text{-game 1} \) for a probabilistic polynomial-time adversary \( T_{1,P} \) is defined as \( \text{game 1} \) with the following changes:

- Before the challenge phase, \( T_{1,P} \) sends \( p \) to the challenger.
- In the challenge phase, the challenger chooses \( b = 0 \) with probability \( p \) and \( b = 1 \) with probability \( 1 - p \).

We call a PSA scheme \( P \)-secure if the probability of every probabilistic polynomial-time adversary \( T_{1,P} \) in winning the above game is \( P \) + \( \text{neg}(\kappa) \). Note that \( \text{game 1} \) is a special case of \( P\text{-game 1} \), where \( P = 1/2 \). We refer to this case as the unbiased version (rather the biased version if \( P > 1/2 \)) of \( P\text{-game 1} \). In the unbiased case, we just drop the dependence on \( P \) and the adversary is not required to send \( p \) to its challenger.

### 4.2 Constructing a PSA adversary using a CDP adversary

#### 4.2.1 Security game for adjacent databases

For showing that a \( P \)-secure PSA scheme is suitable for preserving CDP, we have to construct a successful adversary in \( P\text{-game 1} \) (with a proper choice of \( p \)) using a successful distinguisher for adjacent databases. We define the following \( \text{game 0} \) for a probabilistic polynomial-time adversary \( T_0 \) which is identical to \( \text{game 1} \) with a changed challenge-phase:

**Challenge.** \( T_0 \) chooses \( t^* \in T \) such that no encryption query at \( t^* \) was made. \( T_0 \) queries two adjacent tuples \((d_i^{[0]}), (d_i^{[1]})\) \( i \in U \). For all \( i \in U \) the challenger returns

\[
    c_{i,t^*} = \text{PSAEnc}_s(t^*, x_i^{[b]}),
\]

where \( x_i^{[b]} \) is a noisy version of \( d_i^{[b]} \) \( i \in U \) obtained by some randomised perturbation process.

Now consider the following experiment which we call \( \text{Exp}_1 \). Let \( f_D : \mathcal{D}^n \rightarrow \mathcal{O} \) be a sum query and \( Q : \mathcal{D}^n \rightarrow [0, 1] \) a probability distribution function on \( \mathcal{D}^n \). For simplicity we consider only the case where \( \mathcal{D} \subseteq \mathcal{D} = \mathbb{Z} \). Let \( D_0, D_1 \in \mathcal{D}^n \). Then \( \text{Exp}_1 \) is performed as follows:

- Let \( B_{1/2} \) be a Bernoulli variable with \( \Pr[B_{1/2} = 0] = 1/2 \).
Let \( \hat{X}_1 \) be a random vector with probability distribution function \( Q(x - D_b) \), where \( b \) is a realisation of \( B_{1/2} \).

- Let \( Y = f_B(\hat{X}_1) \).

We now define an experiment \( \text{Exp}_2 \) and afterwards show that it is statistically equivalent to \( \text{Exp}_1 \).

- Let \( Y \) be a random variable as in \( \text{Exp}_1 \).
- Let \( p = \Pr[B_{1/2} = 0 | Y = y] \). Let \( B_p \) be a Bernoulli variable with \( \Pr[B_p = 0] = p \).
- Let \( \hat{X}_2 \) be a random vector with conditional probability function

\[
\Pr[\hat{X}_2 = \hat{D} | B_p = b, Y = y] = \begin{cases} \chi(y) \cdot Q(\hat{D} - D_b) \frac{1}{2p \cdot \Pr[Y = y]}, & \text{if } b = 0 \\ \chi(y) \cdot Q(\hat{D} - D_b) \frac{1}{(1-p) \cdot \Pr[Y = y]}, & \text{if } b = 1. \end{cases}
\]

Here \( b \) is a realisation of \( B_p \) and \( \chi(y) = \chi(z \in \mathcal{O} | z = f_B(\hat{D})) \) denotes the characteristic function of \( \{z \in \mathcal{O} | z = f_B(\hat{D})\} \subset \mathcal{O} \), which is 1 if \( y = f_B(\hat{D}) \) and 0 otherwise. Note that the values \( \chi(y), Q(x), p, \Pr[Y = y] \) for the computation of the conditional probability of \( \hat{X}_2 \) are known.

For showing that \( \text{Exp}_1 \) and \( \text{Exp}_2 \) are statistically equivalent it suffices to show that the joint distributions \( \Pr[B_{1/2}, \hat{X}_1, Y] \) and \( \Pr[B_p, \hat{X}_2, Y] \) are equal.

**Lemma 3.** \( \Pr[B_{1/2}, \hat{X}_1, Y] = \Pr[B_p, \hat{X}_2, Y] \).

**Proof.** We observe that in \( \text{Exp}_1 \), \( \Pr[Y = y | \hat{X}_1 = \hat{D}, B_{1/2} = b] = \Pr[Y = y | \hat{X}_1 = \hat{D}] \). Therefore we have

\[
\Pr[\hat{X}_1 = \hat{D} | B_{1/2} = b, Y = y] = \frac{\Pr[Y = y | \hat{X}_1 = \hat{D}] \cdot Q(\hat{D} - D_b)}{\Pr[Y = y | B_{1/2} = b]} = \frac{\chi(y) \cdot Q(\hat{D} - D_b)}{\Pr[Y = y | B_{1/2} = b]},
\]

which exactly corresponds to the conditional probability of \( \hat{X}_2 \) in \( \text{Exp}_2 \). Thus, we have

\[
\Pr[B_p, \hat{X}_2, Y] = \Pr[Y] \cdot \Pr[B_p] \cdot \Pr[\hat{X}_2 | B_p, Y] = \Pr[Y] \cdot \Pr[B_{1/2} | Y] \cdot \Pr[\hat{X}_1 | B_{1/2}, Y] = \Pr[B_{1/2}, \hat{X}_1, Y].
\]

\( \square \)

Note that Lemma 3 also applies to the marginals of the triples \( (B_{1/2}, \hat{X}_1, Y) \) and \( (B_p, \hat{X}_2, Y) \).
4.2.2 The Reduction

With Lemma 3 in mind, we can show that a successful adversary in \( \text{game 0} \) yields a successful adversary in \( \text{P-game 1} \) for a particular \( P \in \{1/2, 1\} \). Afterwards we show that a successful adversary in \( \text{P-game 1} \) for any \( P \in \{1/2, 1\} \) yields a successful adversary in \( \text{game 1} \).

**Lemma 4.** Let \( \kappa \) be a security parameter. Let \( \mathcal{T}_0 \) be an adversary in \( \text{game 0} \) with advantage \( \mu_0(\kappa) > \text{neg}(\kappa) \). Let \( B_{1/2} \) denote the random variable describing the challenge bit \( b \) in \( \text{game 0} \) and let \( Y \) denote the random variable describing the aggregate of \( (x_i^{[b]})_{i \in U} \). Let \( p \) be the probability of \( B_{1/2} = 0 \) given the choice of \( Y \) and let \( P = \max\{p, 1 - p\} \). Then there exists an adversary \( \mathcal{T}_{1,P} \) in \( \text{P-game 1} \) with advantage \( \mu_{1,P}(\kappa) > \text{neg}(\kappa) \).

**Proof.** We construct a successful adversary \( \mathcal{T}_{1,P} \) in \( \text{P-game 1} \) using \( \mathcal{T}_0 \) as follows:

**Setup.** Receive \( \kappa, pp, T, s \) from the \( \text{P-game 1} \)-challenger and send it to \( \mathcal{T}_0 \).

**Queries.** Receive \( U = \{i_1, \ldots, i_u\} \subseteq \{n\} \) from \( \mathcal{T}_0 \) and send it to the challenger. Forward the obtained response \( (s_i)_{i \in \{n\} \cup U} \) to \( \mathcal{T}_0 \). Forward \( \mathcal{T}_0 \)'s queries \((i, t, d_i)\) with \( i \in U, t \in T, d_i \in D \) to the challenger and forward the obtained response \( c_{i,t} \) to \( \mathcal{T}_0 \).

**Challenge.** \( \mathcal{T}_0 \) chooses \( t^* \in T \) such that no encryption query at \( t^* \) was made and queries two adjacent tuples \( (d_i^{[0]}), (d_i^{[1]})_{i \in U} \). Choose a realisation \( y \) of \( Y \) according to \( \text{Exp}_2 \). Set \( p = \Pr[B_{1/2} = 0 | Y = y] \) and choose \( (x_i^{[a]})_{i \in U} \) with probability \( \Pr[\tilde{X}_2 = (x_i^{[a]})_{i \in U} | B_{1/2} = a, Y = y] \) for \( a = 0, 1 \) according to \( \text{Exp}_2 \). Send \( p, t^*, (x_i^{[0]}, x_i^{[1]})_{i \in U} \) to the challenger. Obtain the response \( (c_{i,t^*})_{i \in U} \) and forward it to \( \mathcal{T}_0 \).

**Queries.** \( \mathcal{T}_0 \) can make the same type of queries as before with the restriction that no encryption query at \( t^* \) can be made.

**Guess.** \( \mathcal{T}_0 \) gives a guess about which database was encrypted. Output the same guess.

The rules of \( \text{P-game 1} \) are preserved since \( \mathcal{T}_{1,P} \) sends two tuples of the same aggregate \( y \) to its challenger. On the other hand, since the ciphertexts generated by the challenger are determined by the challenge bit and the collection \( (x_i^{[b]})_{i \in U} \), the rules of \( \text{game 0} \) are preserved by Lemma 3 (the triple \( (b, (x_i^{[b]})_{i \in U}), y \) is chosen according to \( \text{Exp}_2 \)). Therefore \( \mathcal{T}_{1,P} \) perfectly simulates \( \text{game 0} \) and has the same advantage as \( \mathcal{T}_0 \). \( \square \)

We now show that a secure PSA scheme is also \( P \)-secure for every \( p \in (0,1) \), where \( P = \max\{p, 1 - p\} \).

**Lemma 5.** Let \( \kappa \) be a security parameter. For any \( p \in (0,1) \) let \( \mathcal{T}_{1,P} \) be an adversary in \( \text{P-game 1} \) with advantage \( \mu_{1,P}(\kappa) > \text{neg}(\kappa) \). Then there exists an adversary \( \mathcal{T}_1 \) in \( \text{game 1} \) with advantage \( \mu_1(\kappa) > \text{neg}(\kappa) \).

**Proof.** Given a successful adversary \( \mathcal{T}_{1,P} \) in \( \text{P-game 1} \), we construct a successful adversary \( \mathcal{T}_1 \) in \( \text{game 1} \) as follows:

**Setup.** Receive \( \kappa, pp, T, s \) from the \( \text{game 1} \)-challenger and send it to \( \mathcal{T}_{1,P} \).
Queries. Receive $U \subseteq [n]$ from $T_{1,p}$ and send it to the challenger. Forward the obtained response $(s_i)_{i \in [n]}$ to $T_{1,p}$. Forward $T_{1,p}$’s queries $(i,t,x_i)$ with $i \in U, t \in T, x_i \in \hat{D}$ to the challenger and forward the obtained response $c_{i,t}$ to $T_{1,p}$.

Challenge. $T_{1,p}$ chooses $t^* \in T$ such that no encryption query at $t^*$ was made, sends $p \in (0,1)$ and queries two different tuples $(x_i^{[0]})_{i \in U}, (x_i^{[1]})_{i \in U}$ with $f_{\hat{D}|U}((x_i^{[0]})_{i \in U}) = f_{\hat{D}|U}((x_i^{[1]})_{i \in U})$. Choose a bit $a$ with $\Pr[a = 0] = p, \Pr[a = 1] = 1 - p$ and query $(x_i^{[a]})_{i \in U}, (x_i)_{i \in U}$ to the challenger, where the $x_i$ are chosen uniformly at random from $\hat{D}$ such that $f_{\hat{D}|U}((x_i)_{i \in U}) = f_{\hat{D}|U}((x_i^{[a]})_{i \in U})$. Obtain the response $(c_i,t^*)_{i \in U}$ and forward it to $T_{1,p}$.

Queries. $T_{1,p}$ can make the same type of queries as before with the restriction that no encryption query at $t^*$ can be made.

Guess. $T_{1,p}$ gives a guess about $a$. If the guess is correct, then output 0; if not, output 1.

If $T_{1,p}$ has output the correct guess about $a$ then $T_1$ can say with high confidence that the challenge ciphertexts are the encryptions of $(x_i^{[a]})_{i \in U}$ and therefore outputs 0. On the other hand, if $T_{1,p}$’s guess was not correct, then $T_1$ can say with high confidence that the challenge ciphertexts are the encryptions of the random collection $(x_i)_{i \in U}$ and it outputs 1. Formally:

**Case 1.** Let $(c_i,t^*)_{i \in U} = (\text{PSAEnc}_{s_i}(x_i^{[a]}))_{i \in U}$. Then $T_1$ perfectly simulates $P$-game 1 for $T_{1,p}$ and the distribution of ciphertexts is the same as in $P$-game 1:

\[
\Pr[T_1 \text{ outputs } 0] = p \cdot \Pr[T_{1,p} \text{ outputs } 0 \mid a = 0] + (1 - p) \cdot \Pr[T_{1,p} \text{ outputs } 1 \mid a = 1]
\]

\[
= \Pr[T_{1,p} \text{ wins } P\text{-game 1}]
\]

\[
= P + \mu_1,p(\kappa).
\]

**Case 2.** Let $(c_i,t^*)_{i \in U} = (\text{PSAEnc}_{s_i}(x_i))_{i \in U}$. Then the ciphertexts are random with the constraint that their product is the same as in the first case. The probability that $T_{1,p}$ wins game 1 is at most $P$ and

\[
\Pr[T_1 \text{ outputs } 1] = p \cdot \Pr[T_{1,p} \text{ outputs } 1 \mid a = 0] + (1 - p) \cdot \Pr[T_{1,p} \text{ outputs } 0 \mid a = 1] \cdot (1 - p)
\]

\[
\geq 1 - P.
\]

Finally we obtain that the advantage of $T_1$ in winning game 1 is

\[
\mu_1(\kappa) \geq \frac{1}{2}\mu_1,p(\kappa) > \text{neg}(\kappa).
\]
4.3 Proof of Computational Differential Privacy

We have shown that no probabilistic polynomial-time adversary can win game 0 if the underlying PSA scheme is secure. If the perturbation process in game 0 is \( \epsilon \)-DP-preserving, then the whole construction provides \( \epsilon \)-CDP, as we show now.

**Theorem 6.** Let \( A \) be a mechanism for a query \( f_D : D^n \to O \) which preserves \( \epsilon \)-DP and let \( \Sigma \) be a secure PSA scheme for \( f_D \). Then \( A \) preserves \( \epsilon \)-CDP if it is used for the perturbation process in game 0 instantiated with \( \Sigma \).

**Proof.** Consider again game 1, \( \text{P-game} \ 1 \) and game 0. We first bound the probability \( p = \Pr[B_{1/2} = 0|Y = y] \) for the biased coin in \( \text{P-game} \ 1 \). Since the perturbation process was performed by \( A \), the random variable \( Y \) corresponds to the output of \( A \) and we have

\[
e^{-\epsilon} \cdot \Pr[Y = y|B_{1/2} = 0] \leq \Pr[Y = y|B_{1/2} = 1] \leq e^{\epsilon} \cdot \Pr[Y = y|B_{1/2} = 0].
\]

By the Bayes-formula we get

\[
p_{\text{min}} := \frac{1}{1 + e^{-\epsilon}} \leq p \leq \frac{e^{\epsilon}}{1 + e^{\epsilon}} =: p_{\text{max}}.
\]

Now let \( \mathcal{T} \) be a probabilistic polynomial-time Turing machine. Let \( \mathcal{T}_{1, \nu} \) denote this Turing machine as adversary in the \( \text{P-game} \ 1 \) for any \( \nu = \max\{p, 1-p\} \) with \( p \in [p_{\text{min}}, p_{\text{max}}] \) and let \( \mathcal{T}_1 \) denote the same Turing machine as adversary in game 1. Let finally \( \mathcal{D}_{\text{CDP}} = \mathcal{T}_0 \) denote the same machine as adversary in game 0. Then for \( a = 0, 1 \):

\[
\Pr[\mathcal{D}_{\text{CDP}} = a, B_{1/2} = 1] = \Pr[\mathcal{T}_{1, \nu} = a, B_p = 1] \leq p_{\text{max}} \cdot \Pr[\mathcal{T}_{1, \nu} = a|B_p = 1]
\]

\[
= p_{\text{max}} \cdot \Pr[\mathcal{T}_1 = a|B_{1/2} = 1] \leq p_{\text{max}} \cdot \Pr[\mathcal{T}_1 = a|B_{1/2} = 0] + \text{neg}(\kappa)
\]

\[
= p_{\text{max}} \cdot \frac{p_{\text{min}}}{p_{\text{max}}} \cdot \Pr[\mathcal{T}_{1, \nu} = a, B_p = 0] + \text{neg}(\kappa)
\]

\[
= e^{\epsilon} \cdot \Pr[\mathcal{T}_{1, \nu} = a, B_p = 0] + \text{neg}(\kappa)
\]

\[
= e^{\epsilon} \cdot \Pr[\mathcal{D}_{\text{CDP}} = a, B_{1/2} = 0] + \text{neg}(\kappa).
\]

Equations (1) and (4) hold because of Lemma 5 and Equations (2) and (3) hold because of Lemma 5. It follows that

\[
\Pr[\mathcal{D}_{\text{CDP}} = a|B_{1/2} = 1] \leq e^{\epsilon} \cdot \Pr[\mathcal{D}_{\text{CDP}} = a|B_{1/2} = 0] + \text{neg}(\kappa).
\]

As mentioned at the beginning of this section, we are considering a mechanism which preserves \( \epsilon \)-DP given some event Good. Therefore, also Theorem 6 applies to this mechanism given Good. Accordingly, the mechanism unconditionally preserves \( (\epsilon, \delta) \)-CDP, where \( \delta \) is a bound on the probability that Good does not occur.
Figure 1: Empirical error of the Geometric, Skellam and Binomial mechanisms. The fixed parameters are $\epsilon = 0.1, S(f) = 1, \beta = 0.001$. The left graph shows the mean of the error in absolute value for variable $\delta$ and $\gamma = 1$ over 100 runs, the right graph is for variable $\gamma$ and $\delta = 0.001$.

5 Mechanisms for Differential Privacy

In this section we recall the Geometric mechanism from [27] and the Binomial mechanism from [9] and introduce the Skellam mechanism. Since these mechanisms make use of a discrete probability distribution, they are well-suited for an execution through a secure PSA scheme, thereby preserving computational differential privacy as shown in the last section.

**Theorem 7.** Let $\epsilon > 0$. For all databases $D \in D^n$ the randomised mechanism

$$A(D) := f(D) + Y$$

preserves $(\epsilon, \delta)$-DP with respect to any query $f$ with sensitivity $S(f)$, if $Y$ is distributed according to one of the following probability distributions:

1. $Y \sim \text{Geom}(\lambda)$ with $\lambda = \exp(\epsilon/S(f))$ (and $\delta = 0$) [27],
2. $Y \sim \text{Bin}(n', 1/2)$ with $n' = 64 \cdot S(f)^2 \cdot \log(2/\delta)/\epsilon^2$ [9],
3. $Y \sim \text{Sk}(\mu)$ with

$$\mu = \frac{\log(1/\delta)}{1 - \cosh(\epsilon/S(f)) + (\epsilon/S(f)) \cdot \sinh(\epsilon/S(f))}.$$  

We provide the proof of the third claim in Appendix C. Executing these mechanisms through a PSA scheme requires the use of the known constant $\gamma$ which denotes the a priori estimate of the lower bound on the fraction of non-compromised users. For this case, we provide the accuracy bounds for the aforementioned mechanisms.

---

$^2\text{Sk}(\mu)$ denotes the symmetric Skellam distribution with mean 0 and variance $\mu$. For details, see Appendix C.
Theorem 8. Let $\epsilon > 0, 0 < \delta < 1, S(f) > 0$ and let $0 < \gamma < 1$ be the a priori estimate of the lower bound on the fraction of non-compromised users in the network. By distributing the execution of a perturbation mechanism as described above and using the parameters from Theorem 7, we obtain $(\alpha, \beta)$-accuracy with the following parameters:

1. $\alpha = \frac{4S(f)}{\epsilon} \cdot \sqrt{\frac{1}{\gamma} \cdot \log \left(\frac{1}{\delta}\right) \cdot \log \left(\frac{2}{\beta}\right)}$ for the Geometric mechanism, where $\delta$ bounds the probability that no user has added noise [27],

2. $\alpha = \frac{8\sqrt{2S(f)}}{\epsilon} \cdot \sqrt{\frac{1}{\gamma} \cdot \log \left(\frac{1}{\delta}\right) \cdot \log \left(\frac{2}{\beta}\right)}$ for the Binomial mechanism,

3. $\alpha = \frac{S(f)}{\epsilon} \cdot \left(\frac{1}{\gamma} \cdot \log \left(\frac{1}{\delta}\right) + \log \left(\frac{2}{\beta}\right)\right)$ for the Skellam mechanism.

The second claim can be easily shown using a standard tail bound for the Binomial distribution. The proof of the third claim is provided in Appendix B.

Theorem 8 shows that for constant $\delta, \beta, \gamma$ the errors of the three mechanisms are bounded by $O(S(f)/\epsilon)$ and therefore do not exceed known bounds in the centralised model. As pointed out in Section 2.3, the execution of the Geometric mechanism through a PSA scheme requires each user to generate full noise with a small probability. Complementary, the other two mechanisms allow all users to simply generate noise of small variance. While the accuracy bound of the Geometric is roughly a constant factor smaller than the bound of the Binomial, we obtain a better bound for this second approach using the Skellam mechanism. Specifically, the ratio between the factor $\log(2/\beta) + \log(1/\delta)/\gamma$ in the accuracy of the Skellam mechanism and the factor $\sqrt{\log(2/\beta) \cdot \log(1/\delta)/\gamma}$ in the accuracy of the Geometric mechanism goes to 0 when $\delta$ and $\beta$ go to 0. For example, fix $S(f) = 1, \delta = 0.01, \alpha = 50, \beta = 0.1, \gamma = 1$. Then the Geometric mechanism preserves $(\epsilon, \delta)$-CDP with $\epsilon \approx 0.30$, while the Skellam mechanism preserves $(\epsilon, \delta)$-CDP with $\epsilon \approx 0.15$. An empirical accuracy comparison between the mechanisms is shown in Figure 11. We observe that the error of the Geometric and the Skellam mechanisms have a similar behaviour for both variables $\delta$ and $\gamma$, while the error of the Binomial mechanism is roughly three times larger. Finally, we are able to prove our main result, Theorem 1, which follows from the preceding analyses.

Proof of Theorem 1. The claim follows from Theorem 6 together with Theorem 2 (instantiated with the efficient construction in Example 1) and from Theorem 7 together with Theorem 8.

6 Conclusions

In this work we continued a line of research opened by the work of Shi et al. [27]. By lowering the security definition of a PSA scheme, we were able to prove that a secure scheme (in this sense) can be built upon key-homomorphic weak PRFs. Based on the DDH assumption, we gave an instantiation of a secure PSA scheme. If the plaintext space is large enough, it has a substantially more efficient decryption algorithm than the scheme in [27] at the cost of a slightly less efficient encryption algorithm, and achieves non-adaptive security in the standard model. Using the notion of computational differential privacy, we provided a connection between a secure PSA scheme and a mechanism
preserving differential privacy by showing that a differentially private mechanism preserves computational differential privacy if it is executed through a secure PSA scheme. Moreover, we compared the accuracy of the Geometric, the Binomial and the Skellam mechanisms which preserve differential privacy and are suitable for an execution through a PSA scheme. While the practical performances of the Geometric and the Skellam mechanisms are equally better than the performance of the Binomial mechanism, we were able to provide a slightly better bound for the Skellam mechanism at high privacy levels.

A Proof of Theorem 2

Let game 1 be the security game from Definition 8 instantiated for the PSA scheme of Theorem 2. We need to show that the advantage $\mu(\kappa)$ of a probabilistic polynomial-time adversary $T_1$ in winning this game is negligible in the security parameter $\kappa$. We define the following intermediate game 2 for a probabilistic polynomial-time adversary $T_2$ and then show that winning game 1 is at least as hard as winning game 2.

Setup. The challenger runs the Setup algorithm on input security parameter $\kappa$ and returns public parameters $pp$, time-steps $T$ and secret keys $s, s_1, \ldots, s_n$ with $s = (\mathbf{s}_i^{-1})_{i=1}^n$. It sends $\kappa, pp, T, s$ to $T_2$.

Queries. The challenger flips a random bit $b \in \{0, 1\}$. $T_2$ chooses $U = \{i_1, \ldots, i_u\} \subseteq [n]$ and sends it to the challenger which returns $(s_i)_{i \in [n], U}$. $T_2$ is allowed to query $(i, t, x_i)$ with $i \in U, t \in T, x_i \in D$ and the challenger returns the following: if $b = 0$ it sends $F_{s_i}(t) \cdot \varphi(x_i)$ to $T_2$; if $b = 1$ it chooses

$$h_{1,t}, \ldots, h_{u-1,t} \in_R G'$$

$$h_{u,t} \equiv \prod_{j=1}^u F_{s_{i_j}}(t) \cdot \left(\prod_{j=1}^{u-1} h_{j,t}\right)^{-1}$$

and sends $h_{u,t} \cdot \varphi(x_i)$ to $T_2$.

Challenge. $T_2$ chooses $t^* \in T$ such that no encryption query at $t^*$ was made and queries a tuple $(x_i)_{i \in U}$. If $b = 0$ the challenger sends $(F_{s_i}(t^*) \cdot \varphi(x_i))_{i \in U}$ to $T_2$; if $b = 1$ it chooses

$$h_{1,t^*}, \ldots, h_{u-1,t^*} \in_R G'$$

$$h_{u,t^*} \equiv \prod_{j=1}^u F_{s_{i_j}}(t^*) \cdot \left(\prod_{j=1}^{u-1} h_{j,t^*}\right)^{-1}$$

and sends $(h_{u,t^*} \cdot \varphi(x_i))_{i \in U}$ to $T_2$.

Queries. $T_2$ is allowed to make the same type of queries as before with the restriction that no encryption query at $t^*$ can be made.

Guess. $T_2$ outputs a guess about $b$. 
The adversary wins the game if it correctly guesses $b$.

**Lemma 9.** Let $\kappa$ be a security parameter. Let $T_1$ be an adversary in game 1 with advantage $\mu_1(\kappa) > \text{neg}(\kappa)$. Then there exists an adversary $T_2$ in game 2 with advantage $\mu_2(\kappa) > \text{neg}(\kappa)$.

**Proof.** Given a successful adversary $T_1$ in game 1 we construct a successful adversary $T_2$ in game 2 as follows:

**Setup.** Receive $\kappa, PP, T, s$ from the game 2-challenger and send it to $T_1$.

**Queries.** Flip a random bit $b \in_R \{0, 1\}$. Receive $U = \{i_1, \ldots, i_u\} \subseteq [n]$ from $T_1$ and send it to the challenger. Forward the obtained response $(s_i)_{i \in [n] \setminus U}$ to $T_1$. Forward $T_1$’s queries $(i, t, x_i)$ with $i \in U, t \in T, x_i \in \mathcal{D}$ to the challenger and forward the obtained response $c_{i,t}$ to $T_1$.

**Challenge.** $T_1$ chooses $t^* \in T$ such that no encryption query at $t^*$ was made and queries two different tuples $(x_i^0)_{i \in U}, (x_i^1)_{i \in U}$ with $\sum_{i \in U} x_i^0 = \sum_{i \in U} x_i^1$. Query $(x_i^b)_{i \in U}$ to the challenger. Obtain the response $(c_{i,t^*})_{i \in U}$ and forward it to $T_1$.

**Queries.** $T_1$ can make the same type of queries as before with the restriction that no encryption query at $t^*$ can be made.

**Guess.** $T_1$ gives a guess about $b$. If the guess is correct, then output 0; if not, output 1.

If $T_1$ has output the correct guess about $b$ then $T_2$ can say with high confidence that the challenge ciphertexts were generated using a weak PRF and therefore outputs 0. On the other hand, if $T_1$’s guess was not correct, then $T_2$ can say with high confidence that the challenge ciphertexts were generated using random values and it outputs 1. Formally:

**Case 1.** Let $(c_{i,t^*})_{i \in U} = (F_{s_i}(t^*) \cdot \varphi(x_i^b))_{i \in U}$. Then $T_2$ perfectly simulates game 1 for $T_1$ and the distribution of the ciphertexts is the same as in game 1:

$$\Pr[T_2 \text{ outputs } 0] = \frac{1}{2} (\Pr[T_1 \text{ outputs } 0 \mid b = 0] + \Pr[T_1 \text{ outputs } 1 \mid b = 1])$$

$$= \Pr[T_1 \text{ wins game 1}]$$

$$= \frac{1}{2} + \mu_1(\kappa).$$

**Case 2.** Let $(c_{i,t^*})_{i \in U} = (h_{i,t^*} \cdot \varphi(x_i^b))_{i \in U}$. Then the ciphertexts are random with the constraint

$$\prod_{i \in U} c_{i,t^*} = \prod_{i \in U} F_{s_i}(t^*) \cdot \varphi(x_i^b).$$
such that decryption yields the same sum as in case 1. Because of the perfect security of the one-time pad the probability that $T_1$ wins game 1 is $1/2$ and

$$\Pr[T_2 \text{ outputs } 1] = \frac{1}{2}(\Pr[T_1 \text{ outputs } 1 \mid b = 0] + \Pr[T_1 \text{ outputs } 0 \mid b = 1])$$

$$= \Pr[T_1 \text{ loses game 1}]$$

$$= \frac{1}{2}.$$ 

Finally we obtain that the advantage of $T_2$ in winning game 2 is

$$\mu_2(\kappa) = \frac{1}{2}\mu_1(\kappa) > \text{neg}(\kappa).$$

For a probabilistic polynomial-time adversary $T_3$, we define a new intermediate game 3 out of game 2 by just cancelling the plaintext dependence in each step of game 2, i.e. in the encryption queries and in the challenge, instead of $(i, t, x_i)$ the adversary $T_3$ now just queries $(i, t)$ and the challenger in game 3 sends

$$F_{x_i}(t), \text{ if } b = 0,$$

$$h_{i, t}, \text{ if } b = 1$$

to the adversary $T_3$. The rest remains the same as in game 2.

It is easy to see that if there exists a successful adversary in game 2 then there is also a successful adversary in game 3.

**Lemma 10.** Let $\kappa$ be a security parameter. Let $T_2$ be an adversary in game 2 with advantage $\mu_2(\kappa) > \text{neg}(\kappa)$. Then there exists an adversary $T_3$ in game 3 with advantage $\mu_3(\kappa) > \text{neg}(\kappa)$.

**Remark 2.** For comparison to the proof of adaptive security by Shi et al. [27] we emphasise that in the reduction from Aggregator Obliviousness to an intermediate problem (Proof of Theorem 1 in [27]) an adversary $B$ has to compute the ciphertexts $c_i = g^{x_i} H(t)^{s_i}$ for all users $i \in [n]$ and for all (!) time-steps $t$, since $B$ does not know in advance for which $i \in [n]$ it will have to use the PRF $H(t)^{s_i}$ and for which $i \in [n]$ it will have to use real random values. Thus, $B$ has to program the random oracle $H$ in order to know for all $t$ the corresponding random number $z$ with $H(t) = g^z$ (where $g$ is a generator) for simulating the original Aggregator Obliviousness game. In contrast, in the reduction for our non-adaptive version of Aggregator Obliviousness, it is not necessary to program such an oracle, since the simulating adversary $T_2$ knows in advance the set of non-compromised users and, for all (!) $t$, it can already decide for which $i \in [n]$ it will use the PRF (which in our case is $t^{x_i}$ instead of $H(t)^{s_i}$) and for which $i \in [n]$ it will use a real random value.

In the next step, the problem of distinguishing the weak PRF family

$$\mathcal{F} = \{F_s : M \rightarrow G'\}_{s \in S}$$

from a random function family has to be reduced to the problem of winning game 3. We use a hybrid argument.
Lemma 11. Let $\kappa$ be a security parameter. Let $T_3$ be an adversary in game 3 with advantage $\mu_3(\kappa)$. Then $\mu_3(\kappa) \leq \text{neg}(\kappa)$ if

$$\mathcal{F} = \{ F_s | F_s : M \rightarrow G' \}_{s \in S}$$

is a weak PRF family.

Proof. We define the following sequence of hybrid games, game 3, with $l = 1, \ldots, u - 1$, for a probabilistic polynomial-time adversary $T_3$.

Setup. As in game 3.

Queries. The challenger flips a random bit $b \in R \{0, 1\}$. $T_3$, chooses $U = \{i_1, \ldots, i_u\} \subseteq [n]$ and sends it to the challenger which returns $(s_i)_{i \in [n]} \cup U$. $T_3$ is allowed to query $(i, t)$ with $i \in U, t \in T$ and the challenger returns the following: if $i \notin \{i_1, \ldots, i_{l+b}\}$ it sends $F_{s_i}(t)$ to $T_{3_l}$; if $i \in \{i_1, \ldots, i_{l+b}\}$ it chooses

$$h_{1,t}, \ldots, h_{l-1-b,t} \in R G',$$

$$h_{l+b,t} := \prod_{j=1}^{l+b} F_{s_{i_j}}(t) \cdot \left( \prod_{j=1}^{l-(1-b)} h_{j,t} \right)^{-1}$$

and sends $h_{i,t}$ to $T_{3_l}$.

Challenge. $T_3$ chooses $t^* \in T$ such that no encryption query at $t^*$ was made. The challenger chooses

$$h_{1,t^*}, \ldots, h_{l-1-b,t^*} \in R G',$$

$$h_{l+b,t^*} := \prod_{j=1}^{l+b} F_{s_{i_j}}(t^*) \cdot \left( \prod_{j=1}^{l-(1-b)} h_{j,t^*} \right)^{-1}$$

and sends the following sequence to $T_{3_l}$:

$$(h_{1,t^*}, \ldots, h_{l+b,t^*}, F_{s_{i_{l+b+1}}}(t^*), \ldots, F_{s_{i_u}}(t^*)).$$

Queries. $T_{3_l}$ can make the same type of queries as before with the restriction that no encryption query at $t^*$ can be made.

Guess. $T_{3_l}$ outputs a guess about $b$.

The adversary wins the game if it correctly guesses $b$.

It is easy to see that game 3, with $b = 0$ corresponds to the case $b = 0$ in game 3 and game 3, with $b = 1$ corresponds to the case $b = 1$ in game 3. Moreover the ciphertexts in game 3, with $b = 1$ have the same distribution as the ciphertexts in game 3, with $b = 0$. Therefore

$$\Pr[T_{3_{l+1}} \text{ wins game } 3_{l+1} \mid b = 0] = \Pr[T_{3_l} \text{ loses game } 3_l \mid b = 1].$$
Using a successful adversary $T_{3l}$ in game 3, we construct a successful probabilistic polynomial-time distinguisher $D_{PRF}$ which has access to an oracle

$$O(\cdot) \in R \{F_{s'}(\cdot), \text{rand}(\cdot)\},$$

where

$$F_{s'} : M \rightarrow G'$$

is a weak PRF and

$$\text{rand} : M \rightarrow G'$$

is a real random function. $D_{PRF}$ gets $\kappa$ as input and proceeds as follows.

1. Choose two indices $k_1, k_2 \in [n]$ and guess that $k_1, k_2$ will be the $i_t^{th}$, $i_{t+1}^{th}$ indices in $U$ specified by the adversary $T_{3l}$. This guess will be correct with probability $1/n^2$.

2. Choose $s \in R S, s_i \in R S$ for all $i \in [n]\backslash\{k_1, k_2\}$, generate $pp$ and $T$ with $t \in R M$ for all $t \in T$. Compute $F_s(t)$ for all $t \in T$.

3. Make oracle queries for $t$ and receive $O(t)$ for all $t \in T$.

4. Send $\kappa, pp, T, s$ to $T_{3l}$.

5. **Queries.** Receive $U = \{i_1, \ldots, i_n\} \subseteq [n]$ from $T_{3l}$. If $i_t \neq k_1$ or $i_{t+1} \neq k_2$ then abort. Else send $(s_i)_{i \in [n]} U$ to $T_{3l}$. If $T_{3l}$ queries $(i, t)$ with $i \in U, t \in T$ then return the following: if $i \notin \{i_1, \ldots, i_{t+1}\}$ send $F_{s_i}(t)$ to $T_{3l}$; if $i = i_{t+1} = k_2$ send $O(t)$ to $T_{3l}$; if $i \in \{i_1, \ldots, i_t\}$ choose

$$h_{1, t}, \ldots, h_{l-1, t} \in R G'$$

$$h_{l, t} := \left( F_s(t) \cdot O(t) \cdot \prod_{j=1}^{l-1} h_{j, t} \cdot \prod_{j \in [n] \backslash \{i_1, \ldots, i_{t+1}\}} F_{s_j}(t) \right)^{-1}$$

and send $h_{l, t}$ to $T_{3l}$.

6. **Challenge.** $T_{3l}$ chooses $t^* \in T$ such that no encryption query at $t^*$ was made. Choose

$$h_{1, t^*}, \ldots, h_{l-1, t^*} \in R G'$$

$$h_{l, t^*} := \left( F_s(t^*) \cdot O(t^*) \cdot \prod_{j=1}^{l-1} h_{j, t^*} \cdot \prod_{j \in [n] \backslash \{i_1, \ldots, i_{t+1}\}} F_{s_j}(t^*) \right)^{-1}$$

and send the following sequence to $T_{3l}$:

$$(h_{1, t^*}, \ldots, h_{l, t^*}, O(t^*), F_{s_{l+2}}(t^*), \ldots, F_{s_n}(t^*)).$$

7. **Queries.** $T_{3l}$ can make the same type of queries as before with the restriction that no encryption query at $t^*$ can be made.
8. Guess. \( T_3 \) outputs a guess about whether the \( i_{i+1}^{th} \) element is random or pseudo-random. Output the same guess.\(^3\)

If \( T_3 \) has output the correct guess about whether the \( i_{i+1}^{th} \) element is random or pseudo-random then \( D_{\text{PRF}} \) can distinguish between \( F_s(\cdot) \) and \( \text{rand}(\cdot) \). Now we prove this result formally and show that, in this way, game \( 3_i \) is perfectly simulated by \( T_3 \).

Case 1. Let \( O(\cdot) = F_s(\cdot) \). Define \( s_{i+1} := s' \). Since \( S, M \) are groups, there exists an element \( s_{i} \) with
\[
s_{i} = (s * s_{i+1}^{-1})^{-1}
\]
and for all \( t \in T \):
\[
\left( F_s(t) \cdot F_{s'}(t) \cdot \prod_{i \in [n], \{i_{i+1}\}} F_{s_{i}}(t) \right)^{-1} = \prod_{j=1}^{l} F_{s_{j}}(t).
\]
Then for all \( t \in T \) the value \( h_{t,t} \) is equal to
\[
\left( F_s(t) \cdot \prod_{j=1}^{l-1} h_{j,t} \cdot F_{s'}(t) \cdot \prod_{i \in [n], \{i_{i+1}\}} F_{s_{i}}(t) \right)^{-1} = \prod_{j=1}^{l} F_{s_{j}}(t) \cdot \left( \prod_{j=1}^{l-1} h_{j,t} \right)^{-1}.
\]
Therefore the distribution of the ciphertexts corresponds exactly to the case in game \( 3_i \) with \( b = 0 \).

Case 2. Let \( O(\cdot) = \text{rand}(\cdot) \). Define the random elements
\[
h_{t+1,t} := \text{rand}(t)
\]
for all \( t \in T \). Since \( S, M \) are groups, there exists an element \( s' \in S \) with
\[
s' = (s * s_{i+1}^{-1})^{-1}.
\]
Let \( s_{i} \in R S \) and \( s_{i+1} := s' * s_{i}^{-1} \). Then for all \( t \in T \):
\[
\left( F_s(t) \cdot \prod_{i \in [n], \{i_{i+1}\}} F_{s_{i}}(t) \right)^{-1} = \prod_{j=1}^{l+1} F_{s_{j}}(t)
\]
and the value \( h_{t,t} \) is equal to
\[
\left( F_s(t) \cdot h_{t+1,t} \cdot \prod_{j=1}^{l-1} h_{j,t} \cdot \prod_{i \in [n], \{i_{i+1}\}} F_{s_{i}}(t) \right)^{-1}
\]
\(^3\)Essentially, here the specification of the set of non-compromised users before making any query allows \( D_{\text{PRF}} \) to be consistent with pseudo-random values or real random values in its replies to the queries.
and equivalently
\[ h_{t+1,t} = \prod_{j=1}^{t+1} F_{s_{ij}}(t) \cdot \left( \prod_{j=1}^{t} h_{j,t} \right)^{-1}. \]
Therefore the distribution of the ciphertexts corresponds exactly to the case in game \( 3_i \) with \( b = 1 \).

Without loss of generality, let
\[ \text{Pr}[T_3 \text{ wins game } 3_{i} | b = 0] \geq \text{Pr}[T_3 \text{ loses game } 3_{i} | b = 1]. \]

All in all, we obtain
\[ \text{Pr}[T_3 \text{ wins game } 3_{i} | b = 0] - \text{Pr}[T_3 \text{ loses game } 3_{i} | b = 1] \]
\[ = \text{Pr}[D_{\text{PRF}}^{F_{s_{i}}(\cdot)}(\kappa) = 1 | i_t = k_1, i_{t+1} = k_2] - \text{Pr}[D_{\text{PRF}}^{\text{rand}(\cdot)}(\kappa) = 1 | i_t = k_1, i_{t+1} = k_2] \]
\[ \leq n^2 \cdot (\text{Pr}[D_{\text{PRF}}^{F_{s_{i}}(\cdot)}(\kappa) = 1] - \text{Pr}[D_{\text{PRF}}^{\text{rand}(\cdot)}(\kappa) = 1]) \]
and since \( n \) is polynomial in \( \kappa \), this expression is negligible by the pseudo-randomness of \( F_{s_{i}}(\cdot) \) on uniformly chosen input. Therefore, the advantage of \( T_3 \) in winning game \( 3_{i} \) is negligible.

Finally, by a hybrid argument we have:
\[ \text{Pr}[T_3 \text{ wins game } 3] \]
\[ = \frac{1}{2} (\text{Pr}[T_3 \text{ wins game } 3 | b = 0] + \text{Pr}[T_3 \text{ wins game } 3 | b = 1]) \]
\[ = \frac{1}{2} (\text{Pr}[T_{3_i} \text{ wins game } 3_{i} | b = 0] + \text{Pr}[T_{3_{i-1}} \text{ wins game } 3_{i-1} | b = 1]) \]
\[ = \frac{1}{2} + \frac{1}{2} (\text{Pr}[T_{3_i} \text{ wins game } 3_{i} | b = 0] - \text{Pr}[T_{3_{i-1}} \text{ loses game } 3_{i-1} | b = 1]) \]
\[ = \frac{1}{2} + \frac{1}{2} \sum_{i=1}^{u-1} \text{Pr}[T_{3_i} \text{ wins game } 3_{i} | b = 0] - \text{Pr}[T_{3_i} \text{ loses game } 3_{i} | b = 1] \]
\[ = \frac{1}{2} + \text{neg}(\kappa). \]

We can now complete the proof of Theorem 2.

Proof of Theorem 2 By Lemma 9-11,
\[ \mu_1(\kappa) = 2 \cdot \mu_2(\kappa) = 2 \cdot \mu_3(\kappa) = 2 \cdot (u - 1) \cdot n^2 \cdot \text{neg}(\kappa) < 2 \cdot n^3 \cdot \text{neg}(\kappa) = \text{neg}(\kappa). \]

B The Skellam mechanism

B.1 Preliminaries

As observed before, the distributed noise generation is feasible with a probability distribution function closed under convolution. For this purpose, we recall the Skellam distribution.
Definition 9 (Skellam Distribution [28]). Let $\mu_1, \mu_2 > 0$. A discrete random variable $X$ is drawn according to the Skellam distribution with parameters $\mu_1, \mu_2$ (short: $X \sim \text{Sk} (\mu_1, \mu_2)$) if it has the following probability distribution function $\psi_{\mu_1, \mu_2}: \mathbb{Z} \to \mathbb{R}$:

$$
\psi_{\mu_1, \mu_2}(k) = e^{-(\mu_1+\mu_2)} \left( \frac{\mu_1}{\mu_2} \right)^{k/2} I_k(2\sqrt{\mu_1\mu_2}),
$$

where $I_k$ is the modified Bessel function of the first kind (see pages 374–378 in [21]).

A random variable $X \sim \text{Sk}(\mu_1, \mu_2)$ has variance $\mu_1 + \mu_2$ and can be generated as the difference of two random variables drawn according to the Poisson distribution of mean $\mu_1$ and $\mu_2$, respectively [28]. Note that the Skellam distribution is not generally symmetric. However, we mainly consider the particular case $\mu_1 = \mu_2 = \mu$ and refer to this symmetric distribution as $\text{Sk}(\mu) = \text{Sk}(\mu/2, \mu/2)$.

Lemma 12 (28). Let $X \sim \text{Sk}(\mu_1, \mu_2)$ and $Y \sim \text{Sk}(\mu_3, \mu_4)$ be independent random variables. Then $Z := X + Y$ is distributed according to $\text{Sk}(\mu_1 + \mu_3, \mu_2 + \mu_4)$.

An induction step shows that the sum of $n$ i.i.d. symmetric Skellam random variables with variance $\mu$ is a symmetric Skellam random variable with variance $n\mu$. Suppose that adding symmetric Skellam noise with variance $\mu$ preserves $(\epsilon, \delta)$-DP. Recall that the network is given an a priori known estimate $\gamma$ of the lower bound on the fraction of non-compromised users. We define $\mu_{\text{user}} = \mu/(\gamma n)$ and instruct the users to add symmetric Skellam noise with variance $\mu_{\text{user}}$ to their own data. If compromised users will not add noise, the total noise will be still sufficient to preserve $(\epsilon, \delta)$-DP. For our analysis, we will use the following bound on the ratio of modified Bessel functions of the first kind.

Lemma 13 ([17]). For real $k > 0$ let $I_k(\mu)$ be the modified Bessel function of the first kind and order $k$. Then

$$
\frac{I_k(\mu)}{I_{k+1}(\mu)} < \frac{\mu}{-(k + 1) + \sqrt{(k + 1)^2 + \mu^2}}.
$$

For the privacy analysis of the Skellam mechanism, we need a tail bound on the symmetric Skellam distribution.

Lemma 14. Let $X \sim \text{Sk}(\mu)$ and let $\sigma > 0$. Then, for all $\tau \geq -\sigma \mu$,

$$
\Pr[X > \sigma \mu + \tau] \leq e^{-\mu(1 - \sqrt{1 + \sigma^2 + \sigma \ln(1 + \sigma^2)}) - \tau \ln(1 + \sigma^2)}.
$$

Proof. We use standard techniques from probability theory. Applying Markov’s inequality, for any $t > 0$,

$$
\Pr[X > \sigma \mu + \tau] = \Pr[e^{tX} > e^{t(\sigma \mu + \tau)}] \leq \frac{\mathbb{E}[e^{tX}]}{e^{t(\sigma \mu + \tau)}}.
$$
As shown in [15], for \( X \sim \text{Sk}(\mu) \), the moment generating function of \( X \) is
\[
\mathbb{E}[e^{tX}] = e^{-\mu(1 - \cosh(t))},
\]
where \( \cosh(t) = (e^t + e^{-t})/2 \). Hence, we have
\[
\Pr[X > \sigma \mu + \tau] \leq e^{-\mu(1 - \cosh(t) + t\sigma - t\tau)}.
\]
Fix \( t = \ln(\sigma + \sqrt{1 + \sigma^2}) \). In order to conclude the proof, we observe that \( \cosh(\ln(\sigma + \sqrt{1 + \sigma^2})) = \sqrt{1 + \sigma^2} \).

One can easily verify that, for \( \sigma > 0 \),
\[
1 - \sqrt{1 + \sigma^2} + \sigma \ln(\sigma + \sqrt{1 + \sigma^2}) > 0.
\]

B.2 Analysis of the Skellam mechanism

In this section, we provide a bound on the variance \( \mu \) of the symmetric Skellam distribution (as stated in Theorem 7) that is needed in order to preserve \((\epsilon, \delta)\)-differential privacy and we compute the error that is thus introduced.

Privacy analysis

**Theorem 15.** Let \( \epsilon > 0 \) and let \( 0 < \delta < 1 \). For all databases \( D \in \mathcal{D}^n \) the randomised mechanism
\[
\mathcal{A}_{\text{Sk}}(D) := f(D) + Y
\]
preserves \((\epsilon, \delta)\)-DP with respect to any query \( f \) of sensitivity \( S(f) \), where \( Y \sim \text{Sk}(\mu) \) with
\[
\mu \geq \frac{\log(1/\delta)}{1 - \cosh(\epsilon/S(f)) + (\epsilon/S(f)) \cdot \sinh(\epsilon/S(f))}.
\]

**Proof.** Let \( D_0, D_1 \in \mathcal{D}^n \) be adjacent databases with \( |f(D_0) - f(D_1)| \leq S(f) \). The largest ratio between \( \Pr[\mathcal{A}_{\text{Sk}}(D_0) = R] \) and \( \Pr[\mathcal{A}_{\text{Sk}}(D_1) = R] \) is reached when \( k := R - f(D_0) = R - f(D_1) - S(f) \geq 0 \), where \( R \) is any possible output of \( \mathcal{A}_{\text{Sk}} \). Then, by Lemma [13] for all possible outputs \( R \) of \( \mathcal{A}_{\text{Sk}} \):
\[
\begin{align*}
\frac{\Pr[\mathcal{A}_{\text{Sk}}(D_0) = R]}{\Pr[\mathcal{A}_{\text{Sk}}(D_1) = R]} &= \frac{\Pr[Y = k]}{\Pr[Y = k + S(f)]} \\
&= \prod_{j=1}^{S(f)} \frac{\Pr[Y = k + j - 1]}{\Pr[Y = k + j]} \\
&\leq \prod_{j=1}^{S(f)} \frac{\mu}{(k + j) + \sqrt{(k + j)^2 + \mu^2}} \\
&\leq e^\epsilon.
\end{align*}
\]
Inequality (5) holds if \( k \leq \sinh(\epsilon/S(f)) \cdot \mu - S(f) \), since it implies \( k \leq \sinh(\epsilon/S(f)) \cdot \mu - j \) for all \( j = 1, \ldots, S(f) \) and

\[
\frac{\mu}{-(k + j) + \sqrt{(k + j)^2 + \mu^2}} \leq e^{\epsilon/S(f)}.
\]

Applying Lemma [14] with \( \sigma = \sinh(\epsilon/S(f)) \) and \( \tau = -S(f) \), we get

\[
\Pr[k > \sinh(\epsilon/S(f)) \cdot \mu - S(f)] \leq e^{-\mu \cdot (1 - \cosh(\epsilon/S(f)) + (\epsilon/S(f)) \cdot \sinh(\epsilon/S(f))) + \epsilon}
\]

and this expression is set to be smaller or equal than \( \delta \). This inequality is satisfied if

\[
\mu \geq \frac{\log(1/\delta)}{1 - \cosh(\epsilon/S(f)) + (\epsilon/S(f)) \cdot \sinh(\epsilon/S(f))}.
\]

\[\square\]

**Remark 3.** The bound on \( \mu \) from Theorem [15] is smaller than \( 2 \cdot (S(f)/\epsilon)^2 \cdot \log(1/\delta) \), thus the standard deviation of \( Y \sim S(k) \) is linear in \( S(f)/\epsilon \) (for constant \( \delta \)).

**Accuracy analysis**

**Theorem 16.** Let \( \epsilon > 0 \) and \( 0 < \delta < 1 \). Then for all \( 0 < \beta < 1 \) the mechanism specified in Theorem [15] has \((\alpha, \beta)\)-accuracy, where

\[
\alpha = \frac{S(f)}{\epsilon} \cdot \left( \log \left( \frac{2}{\beta} \right) + \log \left( \frac{1}{\delta} \right) \right).
\]

**Proof.** Let \( \mu = \frac{\log(1/\delta)}{1 - \cosh(\epsilon/S(f)) + (\epsilon/S(f)) \cdot \sinh(\epsilon/S(f))} \) be the bound on the variance for the Skellam mechanism provided in Theorem [15]. Now, as in the proof of Lemma [14] for \( \alpha' > 0 \),

\[
\Pr[|N| > \alpha'] = 2 \cdot \Pr[N > \alpha'] \leq 2 \cdot e^{-\mu \cdot (1 - \cosh(\epsilon/S(f)) - (\epsilon/S(f)) \cdot \alpha')}
\]

and this expression is set to be equal to \( \beta \). Solving this equality for \( \alpha' \) yields

\[
\alpha' = \frac{S(f)}{\epsilon} \cdot \left( \log \left( \frac{2}{\beta} \right) + \cosh \left( \frac{\epsilon}{S(f)} \right) - 1 \right) \cdot \mu
\]

\[
\leq \frac{S(f)}{\epsilon} \cdot \left( \log \left( \frac{2}{\beta} \right) + \log \left( \frac{1}{\delta} \right) \right)
\]

\[= \alpha.\]
For the distributed noise generation, each single user adds symmetric Skellam noise with variance \( \mu_{\text{user}} = \mu/(\gamma n) \) to her data. The worst case for accuracy is when all \( n \) users add noise, thus the total noise \( N \) is a symmetric Skellam variable with variance \( \mu/\gamma \) and the accuracy becomes

\[
\alpha = \frac{S(f)}{\epsilon} \cdot \left( \log \left( \frac{2}{\beta} \right) + \frac{1}{\gamma} \cdot \log \left( \frac{1}{\delta} \right) \right),
\]

proving the third claim of Theorem 8.

References

[1] Milton Abramowitz and Irene A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover Publications, 1964.

[2] Gergely Ács and Claude Castelluccia. I have a dream!: Differentially private smart metering. In *Proc. of IH ’11*, pages 118–132, 2011.

[3] Amos Beimel, Kobbi Nissim, and Eran Omri. Distributed private data analysis: Simultaneously solving how and what. In *Proc. of CRYPTO ’08*, pages 451–468, 2008.

[4] Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to non-interactive database privacy. In *Proc. STOC ’08*, pages 609–618, 2008.

[5] Claude Castelluccia. Efficient aggregation of encrypted data in wireless sensor networks. In *Proc. of In MobiQuitous ’05*, pages 109–117, 2005.

[6] Claude Castelluccia, Aldar C-F. Chan, Einar Mykletun, and Gene Tsudik. Efficient and provably secure aggregation of encrypted data in wireless sensor networks. *ACM Trans. Sen. Netw.*, 5(3):20:1–20:36, 2009.

[7] T.-H. Hubert Chan, Elaine Shi, and Dawn Song. Privacy-preserving stream aggregation with fault tolerance. In *Proc. of FC ’12*, pages 200–214, 2012.

[8] Cynthia Dwork. Differential privacy: A survey of results. In *Proc. of TAMC ’08*, pages 1–19, 2008.

[9] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Proc. of EUROCRYPT ’06*, pages 486–503.

[10] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Proc. of TCC ’06*, pages 265–284, 2006.

[11] Fabienne Eigner, Aniket Kate, Matteo Maffei, Francesca Pampaloni, and Ivan Pryvalov. Differentially private data aggregation with optimal utility. In *Proc. of ACSAC ’14*, pages 316–325, 2014.

[12] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, August 1986.
[13] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In Proc. of STOC ’87, pages 218–229, 1987.

[14] Slawomir Goryczka, Li Xiong, and Vaidy Sunderam. Secure multiparty aggregation with differential privacy: A comparative study. In Proc. of EDBT ’13, pages 155–163, 2013.

[15] Joseph Oscar Irwin. The frequency distribution of the difference between two independent variates following the same poisson distribution. Journal of the Royal Statistical Society, 100(3):415–416, 1937.

[16] Marc Joye and Benoît Libert. A scalable scheme for privacy-preserving aggregation of time-series data. In Proc. of FC ’13, pages 111–125, 2013.

[17] Andrea Laforgia and Pierpaolo Natalini. Some inequalities for modified bessel functions. Journal of Inequalities and Applications, 2010(1), 2010.

[18] Qinghua Li and Guohong Cao. Efficient and privacy-preserving data aggregation in mobile sensing. In Proc. of ICNP ’12, pages 1–10, 2012.

[19] Qinghua Li and Guohong Cao. Efficient privacy-preserving stream aggregation in mobile sensing with low aggregation error. In Proc. of PETS ’13, pages 60–81, 2013.

[20] Yehuda Lindell and Benny Pinkas. Secure multiparty computation for privacy-preserving data mining. Journal of Privacy and Confidentiality, 1(1):5, 2009.

[21] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In Proc. of FOCS ’07.

[22] Ilya Mironov, Omkant Pandey, Omer Reingold, and Salil Vadhan. Computational differential privacy. In Proc. of CRYPTO ’09, pages 126–142, 2009.

[23] Moni Naor, Benny Pinkas, and Omer Reingold. Distributed pseudo-random functions and kdes. In Proc. of EUROCRYPT ’99, pages 327–346, 1999.

[24] Moni Naor and Omer Reingold. Synthesizers and their application to the parallel construction of pseudo-random functions. In Proc. of FOCS ’95, pages 170–181, 1995.

[25] Vibhor Rastogi and Suman Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In Proc. of SIGMOD ’10, pages 735–746, 2010.

[26] Eleanor G. Rieffel, Jacob T. Biehl, William van Melle, and Adam J. Lee. Secured histories: computing group statistics on encrypted data while preserving individual privacy. CoRR, abs/1012.2152, 2010.

[27] Elaine Shi, T.-H. Hubert Chan, Eleanor G. Rieffel, Richard Chow, and Dawn Song. Privacy-preserving aggregation of time-series data. In Proc. of NDSS ’11, 2011.
[28] John Gordon Skellam. The frequency distribution of the difference between two poisson variates belonging to different populations. *Journal of the Royal Statistical Society*, 109(3):296, 1946.

[29] Jonathan Ullman and Salil Vadhan. Pcps and the hardness of generating private synthetic data. In *Proc. of TCC ’11*, pages 400–416, 2011.