Aggregation of Time-Series Data under Differential Privacy *

Filipp Valovich
Email: filipp.valovich@rub.de

Horst Görtz Institute for IT Security
Faculty of Mathematics
Ruhr-Universität Bochum, Universitätsstraße 150, 44801 Bochum, Germany

Abstract. In this work, we investigate the problem of statistical data analysis while preserving user privacy in the distributed and semi-honest setting. Particularly, we study properties of Private Stream Aggregation (PSA) schemes, first introduced by Shi et al. in 2011. A PSA scheme is a secure multiparty protocol for the aggregation of time-series data in a distributed network with a minimal communication cost. We show that in the non-adaptive query model, secure PSA schemes can be built upon any key-homomorphic weak pseudo-random function (PRF) and we provide a tighter security reduction. In contrast to the aforementioned work, this means that our security definition can be achieved in the standard model. In addition, we give two computationally efficient instantiations of this theoretic result. The security of the first instantiation comes from a key-homomorphic weak PRF based on the Decisional Diffie-Hellman problem and the security of the second one comes from a weak PRF based on the Decisional Learning with Errors problem. Moreover, due to the use of discrete Gaussian noise, the second construction inherently maintains a mechanism that preserves (ϵ, δ)-differential privacy in the final data-aggregate. A consequent feature of the constructed protocol is the use of the same noise for security and for differential privacy. As a result, we obtain an efficient prospective post-quantum PSA scheme for differentially private data analysis in the distributed model.

Keywords: Aggregator Obliviousness, Post-Quantum Cryptography, Differential Privacy

1 Introduction

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* (see [4], [8], [10], [17]) who collects data in the clear, aggregates and perturbs it properly (e.g. by adding Laplace-distributed

* The research was supported by the DFG Research Training Group GRK 1817/1

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 and simultaneously high accuracy of the analysis is maintained. In this work, we study how to preserve differential privacy when we cannot rely on a trusted curator. In this *distributed setting*, the users have to send their own 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. To this end, a Private Stream Aggregation (PSA) scheme can be deployed. A PSA scheme is a cryptographic protocol enabling each user of the network to securely send encrypted time-series data to an untrusted aggregator requiring each user to send exactly one message per time-step. 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 the differential privacy of the final aggregate is preserved *and* high accuracy is guaranteed. In this framework, the results of this work are as follows: first we show that a secure PSA scheme can be built upon any *key-homomorphic weak pseudo-random function*. From this result, we construct a PSA scheme based on the *Decisional Diffie-Hellman* (DDH) assumption. Moreover, we construct a PSA scheme based on the *Decisional Learning with Errors* (DLWE) assumption that is prospectively secure in the post-quantum world and automatically provides differential privacy to users.

Related work. The concept of PSA was introduced by Shi et al. [25], where a PSA scheme for sum-queries was provided and shown to be secure under the DDH assumption. 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. In contrast, our schemes are secure in the *standard model* and can efficiently decrypt the aggregate, even if the users' data consist of large numbers. In [13], a PSA scheme was provided that is secure in the random oracle model based on the Decisional Composite Residuosity (DCR) assumption. As a result, a factor which is cubic in the number of users can be removed in the security reduction. However, this scheme involves a semi-trusted party for setting some public parameters. In our work, we provide instantiations of our *generic* PSA construction which rely on the DDH assumption and on the DLWE assumption. While in our generic security reduction we cannot avoid a *linear* factor in the number of users, our construction *does not* involve any trusted party and has security guarantees in the standard model. In a subsequent work [3], a generalisation of the scheme from [13] is obtained based on smooth projective hash functions [7]. This allows the construction of secure protocols based on various hardness assumptions (such as the k-LIN assumption). However, the dependencies on a semi-trusted party (for most of the instantiations) and on a random oracle remain.

2 Preliminaries

2.1 Model

We consider a distributed network of users with sensitive data stored in a database. In a distributed network, the users perform computations on their own and do not trust other users or other parties outside the network. More specifically, we assume the existence of an aggregator with the aim to analyse the data in the database. The users are asked to participate in some statistical analyses but do not trust the data aggregator (or analyst), who is assumed to be honest but curious and therefore *corrupted*. In this so-called *semi-honest* model, the users do not provide their own sensitive data in the clear, since they want to preserve the privacy of their data. On the other hand, the untrusted analyst wants to compute some (pre-defined) statistics over these data sets and moreover will use any auxiliary obtained information in order to deduce some more information about the individual users' data. Despite that, and opposed to the *malicious* model, where the corrupted parties may execute *any* behaviour, the analyst will honestly follow the rules of the network.
Moreover, the users perform computations independently and communicate solely and independently with the untrusted analyst. We also assume that the analyst may corrupt users of the network in order to compromise the privacy of the other participants. Uncorrupted users honestly follow the rules of the network and want to release useful information about their data (with respect to particular statistical database queries by the analyst), while preserving the privacy of their data. The remaining users are assumed to be corrupted and following the rules of the network but aiming at violating the privacy of uncorrupted users. For that purpose, these users form a coalition with the analyst and send auxiliary information to the analyst, e.g. their own data in the clear. Therefore, the members of the coalition are allowed to communicate among each other and with the analyst at any time.
We consider all parties to have only limited computational ressources, i.e. we consider only algorithms with a running time that is polynomial in some complexity parameter.
The untrusted data analyst wants to analyse the users' data by means of time-series queries and aims at obtaining answers as accurate as possible. More specifically, assume that the users have a series of data items belonging to a data universe $\mathcal{D}$. For a sequence of time-steps $t \in T$, where T is a discrete time period, the analyst sends queries which are answered by the users in a distributed manner. Each query is modelled as a function $f : \mathcal{D}^n \to \mathcal{O}$ for a finite or countably infinite set of possible outputs (i.e. answers to the query) $\mathcal{O}$. We consider only sum-queries in this work.
For computing the answers to the aggregator's queries, a special cryptographic protocol, the Private Stream Aggregation (PSA) scheme, is executed by *all* users. In contrast to common secure multi-party techniques (see [12] and [15]), this protocol requires each user to send only one message per time-series query to the analyst.

2.2 Cryptographic hardness assumptions and pseudo-random functions

In the following, we describe some cryptographic hardness assumptions that will be the basis for two weak pseudo-random functions that we will use for the construction of secure PSA schemes.

The Decisional Diffie-Hellman Assumption. The Decisional Diffie-Hellman (DDH) assumption underlies one of our constructions. It is related to the discrete logarithm (dlog) assumption: it says that no probabilistic polynomial time (ppt) algorithm is able to find $x \in \mathbb{Z}_q$ given g^x for a generator g of a finite cyclic group of order q. Under the DDH assumption, a ppt algorithm cannot distinguish g^{xy} from a random element in the group. The DDH assumption implies the dlog assumption, since by solving the discrete logarithm one can easily distinguish g^z from g^{xy}.

The Learning with Errors Assumption. As an instance of the Learning with Errors (LWE) problem, we are given a uniformly distributed matrix $\mathbf{A} \in \mathbb{Z}_q^{\lambda \times \kappa}$ and a noisy codeword $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{e} \in \mathbb{Z}_q^{\lambda}$ with an error term $\mathbf{e} \in \mathbb{Z}_q^{\lambda}$ sampled according to a proper known probability distribution χ^{λ} and an unknown uniform $\mathbf{x} \in \mathbb{Z}_q^{\kappa}$. The task is to find the correct vector $\mathbf{x}$. Without the error term, the task would simply be to find the solution to a system of linear equations. Thus, the error term is crucial for the hardness of this problem. In the decisional version of this problem (DLWE problem), we are given $(\mathbf{A}, \mathbf{y})$ and have to decide whether $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{e}$ or $\mathbf{y}$ is a uniformly distributed vector in $\mathbb{Z}_q^{\lambda}$.
Regev [23] established the average-case-hardness of the search problem by the construction of an efficient quantum algorithm for worst-case lattice problems using an efficient solver for LWE, if the errors follow a *discrete Gaussian* distribution $D(\nu)$, where $X \sim D(\nu)$ iff $\Pr[X = x] = (1/c_\nu) \cdot \exp(-\pi x^2/\nu)$ with $c_\nu = \sum_x \exp(-\pi x^2/\nu)$.

Theorem 1 (Worst-to-Average Case [23]) *Let κ be a security parameter and let $q = q(\kappa)$ be a modulus, let $\alpha = \alpha(\kappa) \in (0, 1)$ be such that $\alpha q > 2\sqrt{\kappa}$. If there exists a ppt algorithm solving the* LWE *problem with errors distributed according to $D((\alpha q)^2/(2\pi))$ with more than negligible probability, then there exists an efficient quantum algorithm that approximates the decisional shortest vector problem (*GapSVP*) and the shortest independent vectors problem (*SIVP*) to within $\tilde{O}(\kappa/\alpha)$ in the worst case.*

Opposed to this quantum reduction, Peikert [22] provided a classical reduction. However, Regev's result suffices for our purposes. Micciancio and Mol [18] provided a sample preserving search-to-decision reduction that works for any error distribution χ, showing the equivalence in the average case.

Theorem 2 (Search-to-Decision [18]) *Let κ be a security parameter, $q = q(\kappa) = \mathsf{poly}(\kappa)$ a prime modulus and let χ be any distribution on $\mathbb{Z}_q$. Assume*

there exists a ppt distinguisher that solves the DLWE *problem with more than negligible success-probability, then there exists a ppt adversary that solves the* LWE *problem with more than negligible success-probability (both with error distribution χ).*

Pseudo-Random Functions. A PRF family is a collection of efficiently computable functions that cannot be distinguished from random functions on *arbitrarily* distributed input by any ppt algorithm with black-box access to the function and with more than negligible probability. Constructing a PRF is possible e.g. by assuming the existence of a random oracle, i.e. a random black-box function that obtains an arbitrarily distributed input and outputs something uniformly random (but always the same for the same input). If the range of the random oracle is large enough, each input will have a unique (random) output with overwhelming probability. Then the output of the random oracle serves as input to a *weak* PRF, i.e. a function that takes *uniformly distributed* values as input and outputs pseudo-random values.

Definition 1 ((Weak) PRF [21]) *Let κ be a security parameter. Let A, B, C be sets with sizes parameterised by κ. A family of functions $\mathcal{F} = \{\mathsf{F}_a \mid \mathsf{F}_a : B \to C\}_{a \in A}$ is called a* (respectively weak) pseudo-random function (PRF) family*, if for all ppt algorithms $\mathcal{D}_{PRF}^{\mathcal{O}(\cdot)}$ with oracle access to $\mathcal{O}(\cdot) \in \{\mathsf{F}_a(\cdot), \mathsf{rand}(\cdot)\}$, on any polynomial number of arbitrarily chosen (respectively uniform and given) inputs, we have $|\Pr[\mathcal{D}_{PRF}^{\mathsf{F}_a(\cdot)}(\kappa) = 1] - \Pr[\mathcal{D}_{PRF}^{\mathsf{rand}(\cdot)}(\kappa) = 1]| \leqslant \mathsf{neg}(\kappa)$, where $a \leftarrow \mathcal{U}(A)$ and $\mathsf{rand} \in \{f \mid f : B \to C\}$ is a random mapping from B to C.*

2.3 Private Stream Aggregation

In this section, we define Private Stream Aggregation (PSA) and provide two security definitions. The notion of PSA was introduced by Shi et al. [25].

The definition of Private Stream Aggregation. A PSA scheme is a protocol for safe distributed time-series data transfer which enables the receiver (here: the untrusted analyst) to learn nothing else than the sums $\sum_{i=1}^{n} x_{i,j}$ for $j = 1, 2, \ldots$, where $x_{i,j}$ is the value of the ith participant in time-step j and n is the number of participants (or users). Such a scheme needs a key exchange protocol for all n users together with the analyst as a precomputation (e.g. using multi-party techniques), and requires each user to send exactly one message in each time-step $j = 1, 2, \ldots$.

Definition 2 (Private Stream Aggregation [25]) *Let κ be a security parameter, $\mathcal{D}$ a set and $n = poly(\kappa)$, $\lambda = poly(\kappa)$. A* Private Stream Aggregation *(PSA) scheme $\Sigma = (\mathsf{Setup}, \mathsf{PSAEnc}, \mathsf{PSADec})$ is defined by three ppt algorithms:*

Setup*: $(\mathsf{pp}, T, s_0, s_1, \ldots, s_n) \leftarrow \mathsf{Setup}(1^\kappa)$ with public parameters pp, $T = \{t_1, \ldots, t_\lambda\}$ and secret keys s_i for all $i = 1, \ldots, n$.*

PSAEnc: *For $t_j \in T$ and all $i = 1, \dots, n$: $c_{i,j} \leftarrow \mathsf{PSAEnc}_{s_i}(t_j, x_{i,j})$ for $x_{i,j} \in \mathcal{D}$.*
PSADec: *Compute $\sum_{i=1}^{n} x'_{i,j} = \mathsf{PSADec}_{s_0}(t_j, c_{1,j}, \dots, c_{n,j})$ for $t_j \in T$ and ciphers $c_{1,j}, \dots, c_{n,j}$. For all $t_j \in T$ and $x_{1,j}, \dots, x_{n,j} \in \mathcal{D}$ the following holds:*

$$\mathsf{PSADec}_{s_0}(t_j, \mathsf{PSAEnc}_{s_1}(t_j, x_{1,j}), \dots, \mathsf{PSAEnc}_{s_n}(t_j, x_{n,j})) = \sum_{i=1}^{n} x_{i,j}.$$

The system parameters pp are public and constant for all t_j with the implicit understanding that they are used in Σ. Every user encrypts her values $x_{i,j}$ with her own secret key s_i and sends the ciphertext to the untrusted analyst. If the analyst receives the ciphertexts of *all* users for some t_j, it can compute the aggregate of the users' data using the decryption key s_0.

Security of Private Stream Aggregation. Our model allows an attacker to corrupt users. It can obtain auxiliary information about the values of users or their secret keys. Even then a secure PSA scheme should release no more information than the aggregates of the uncorrupted users' values. The difference between the following two security definitions is whether the attacker can corrupt users adaptively or not.

Definition 3 (Adaptive (resp. non-adaptive) Aggregator Obliviousness) *Let κ be a security parameter. Let $\mathcal{T}$ be a ppt adversary for a PSA scheme $\Sigma = (\mathsf{Setup}, \mathsf{PSAEnc}, \mathsf{PSADec})$ and let $\mathcal{D}$ be a set. We define a security game between a challenger and the adversary $\mathcal{T}$.*

Setup. *The challenger runs the* Setup *algorithm on input security parameter κ and returns public parameters pp, public encryption parameters T with $|T| = \lambda = poly(\kappa)$ and secret keys $s_0, s_1, \dots, s_n$. It sends $\kappa, \mathsf{pp}, T, s_0$ to $\mathcal{T}$.*
Queries. *The challenger flips a random bit $b \leftarrow_R \{0,1\}$. $\mathcal{T}$ is allowed to query $(i, t_j, x_{i,j})$ with $i \in \{1, \dots, n\}, t_j \in T, x_{i,j} \in \mathcal{D}$ and the challenger returns $c_{i,j} \leftarrow \mathsf{PSAEnc}_{s_i}(t_j, x_{i,j})$. Moreover, $\mathcal{T}$ is allowed to make compromise queries $i \in \{1, \dots, n\}$ and the challenger returns s_i.*
Challenge. *$\mathcal{T}$ chooses $U \subseteq \{1, \dots, n\}$ such that no compromise query for $i \in U$ was made and sends U to the challenger. $\mathcal{T}$ chooses $t_{j*} \in T$ such that no encryption query with t_{j*} was made. (If there is no such t_{j*} then the challenger simply aborts.) $\mathcal{T}$ queries two different tuples $(x^{[0]}_{i,j*})_{i \in U}, (x^{[1]}_{i,j*})_{i \in U}$ with $\sum_{i \in U} x^{[0]}_{i,j*} = \sum_{i \in U} x^{[1]}_{i,j*}$. For all $i \in U$ the challenger returns $c_{i,j*} \leftarrow \mathsf{PSAEnc}_{s_i}(t_{j*}, x^{[b]}_{i,j*})$.*
Queries. *$\mathcal{T}$ is allowed to make the same type of queries as before restricted to encryption queries with $t_j \neq t_{j*}$ and compromise queries for $i \notin U$.*
Guess. *$\mathcal{T}$ outputs a guess about b.*

The adversary's probability to win the game (i.e. to guess b correctly) is $1/2 + \nu(\kappa)$. A PSA scheme is adaptively aggregator oblivious *or achieves* adaptive Aggregator Obliviousness *(*AO1*) if there is no ppt adversary $\mathcal{T}$ with more than*

negligible advantage $\nu(\kappa) > \mathsf{neg}(\kappa)$ *in winning the above game. A PSA scheme is* non-adaptively aggregator oblivious *or achieves* non-adaptive Aggregator Obliviousness *(*AO2*), if there is no ppt adversary* $\mathcal{T}$ *with advantage* $\nu(\kappa) > \mathsf{neg}(\kappa)$ *in winning a modified game, where the set* U *is already specified by* $\mathcal{T}$ *in the beginning of the first Queries phase, compromise queries are made for all* $i \notin U$ *and encryption queries can only be made for* $i \in U$*. A PSA scheme is* secure *if it achieves either* AO1 *or* AO2*.*

Encryption queries are made only for $i \in U$, since knowing the secret key for all $i \notin U$ the adversary can encrypt a value autonomously. If encryption queries in time-step t_{j*} were allowed, then no deterministic scheme would be secure. The adversary $\mathcal{T}$ can determine the original data of all $i \notin U$, since it knows $(s_i)_{i \notin U}$. Then $\mathcal{T}$ can compute the sum $\sum_{i \in U} x_{i,j} = \mathsf{PSADec}_{s_0}(t_j, c_{1,j}, \ldots, c_{n,j}) - \sum_{i \notin U} x_{i,j}$ of the uncorrupted users' values. If there is a user's cipher which $\mathcal{T}$ does not receive, then it cannot compute the sum for the corresponding t_j. AO2 differs from AO1 in that the first one requires the adversary to specify the set U of uncorrupted users *before* making any query, i.e. it does not allow the adversary to determine U adaptively. The notion of AO1 was introduced in [25].

Feasibility of AO1. In the random oracle model, we can achieve the stronger notion AO1 for some constructions. For example, in [25] the following PSA scheme for sum-queries was proposed. It achieves AO1 based on the DDH assumption.

Example 1 ([25]) **Setup***: The public parameters are a prime* p*, some generator* $g \in \mathbb{Z}_p^*$ *and a hash function* $H : T \to \mathbb{Z}_p^*$ *modelled as a random oracle. The secret keys are* $s_0, \ldots, s_n \leftarrow \mathcal{U}(\mathbb{Z}_p)$ *with* $\sum_{i=0}^{n} s_i \equiv 0 \bmod p-1$*.*

PSAEnc*: For* $t_j \in T$ *and all* $i = 1, \ldots, n$*, encrypt* $x_{i,j} \in \mathbb{Z}_p$ *by* $c_{i,j} \leftarrow g^{x_{i,j}} \cdot H(t_j)^{s_i}$*.*

PSADec*: For* $t_j \in T$ *and ciphers* $c_{1,j}, \ldots, c_{n,j}$*, compute the discrete logarithm of* $V_j \leftarrow H(t_j)^{s_0} \cdot \prod_{i=1}^{n} c_{i,j}$*. If the* $c_{i,j}$ *are encryptions of the* $x_{i,j}$*, then* $V_j = g^{\sum_{i=1}^{n} x_{i,j}}$ *and computing the discrete logarithm of* V_j *to the base* g *yields the desired output.*

Note that the computation of the discrete logarithm may be inefficient, if the range to search in is super-polynomially large in the security parameter. In Section 4.1, we provide our DDH-based example of a PSA scheme that achieves AO1 in the random oracle model and AO2 in the standard model and has an efficient decryption algorithm even if the plaintext space is super-polynomially large.

2.4 Differential Privacy

We consider a database as an element $D \in \mathcal{D}^n$ with data universe $\mathcal{D}$ and number of users n. We will always assume that a privacy-preserving mechanism for

analysing D is applied in the distributed setting. Differential privacy 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 (databases differing in one entry only) leads to close distributions of the outputs.

Definition 4 (Differential Privacy [10]) *Let $\mathcal{R}$ be a set and let $n \in \mathbb{N}$. A randomised mechanism $\mathcal{A} : \mathcal{D}^n \to \mathcal{R}$ preserves* (ϵ, δ)-differential privacy *(short:* DP*), if for all adjacent databases $D_0, D_1 \in \mathcal{D}^n$ and all measurable $R \subseteq \mathcal{R}$:*

$$\Pr[\mathcal{A}(D_0) \in R] \leqslant e^{\epsilon} \cdot \Pr[\mathcal{A}(D_1) \in R] + \delta.$$

The probability space is defined over the randomness of $\mathcal{A}$.

Thus, the presence or absence of a single user does not affect the probability of any outcome by too much. The aim of the analyst is to obtain information from the database. Therefore it processes queries to the database which are answered while preserving DP. In the literature, there are well-established mechanisms for preserving DP (see [10] and [17]).[1] In order to privately evaluate a query, these mechanisms draw error terms according to some distribution depending on the query's global sensitivity. For any $D \in \mathcal{D}^n$, the global sensitivity $S(f)$ of a query $f : \mathcal{D}^n \to \mathbb{R}$ is defined as the maximum change (in terms of the L_1-norm) of $f(D)$, which can be produced by a change of one entry (i.e. the absence of one user) in D. In particular, we will consider sum-queries $f_{\mathcal{D}} : \mathcal{D}^n \to \mathbb{Z}$ or $f_{\mathcal{D}} : \mathcal{D}^n \to [-m', m']$ for some integer m' defined as $f_{\mathcal{D}}(D) := \sum_{i=1}^{n} d_i$, for $D = (d_1, \ldots, d_n) \in \mathcal{D}^n$ and $\mathcal{D} \subseteq \mathbb{Z}$. If the entries in D are bounded by m, then $S(f_{\mathcal{D}}) \leqslant m$. For measuring how well the output of a mechanism $\mathcal{A}$ estimates the real data with respect to a particular query f (mapping into a metric space), we use the notion of (α, β)-accuracy, defined as $\Pr[|\mathcal{A}(D) - f(D)| \leqslant \alpha] \geqslant 1 - \beta$.

The introduction of differential privacy in 2006 was due to the incapability of cryptography to handle data secrecy and analysability at the same time. It is a mathematically well-founded notion for *privacy-preserving data analysis*. Before that, notions of *syntactic anonymity* for *privacy-preserving data publishing*, like k-anonymity [24], were considered by researchers, where the published anonymised data can be used for data analysis tasks. This model asks that every published record is indistinguishable from $k-1$ other records. Its extensions like l-diversity [16] or t-closeness [14] gain more privacy (at the cost of loss in data management effectiveness) by introducing equivalence classes for data and attributes (to also prevent attribute disclosure) and thus reducing the representation granularity. However, this model is vulnarable to certain attacks, that extract *belief probabilities* from the published data. Moreover, applying this model on high-dimensional data leads to a *degradation of data quality*. As noticed in [6], although these issues are not unsolvable, they led to a stronger research focus

[1] These mechanisms work in the centralised setting, where a *trusted curator* sees the full database in the clear and perturbs it properly.

towards differential privacy. Indeed, also differential privacy suffers from some (practical) limitations, like the difficulty to determine a privacy-budget and to compute the exact global sensitivity of the performed data analysis in advance. This leads to excessive perturbation of the correct analysis. Opposed to syntactic anonymity, in differential privacy the true and the perturbed analyses are *probabilistically* correlated, which leads to high uncertainty. Moreover the application of differential privacy makes the assumption of independent data owners, which is not necessary in syntactic anonymity. Therefore the influence of a single individual on the other database participants may be underestimated.
Hence, we see that both approaches and their extensions have advantages and disadvantages while being applied in different tasks and leading to interesting and relevant research challenges.

3 Feasibility of AO2

In this section, we show that a PSA scheme achieving AO2 can be built upon a key-homomorphic weak PRF.

Theorem 3 (Weak PRF gives secure PSA scheme) *Let κ be a security parameter, and $m, n \in \mathbb{N}$ with $\log(m) = \mathsf{poly}(\kappa), n = \mathsf{poly}(\kappa)$. Let $(G, \cdot), (S, *)$ be finite groups and $G' \subseteq G$. For some finite set M, let $\mathcal{F} = \{\mathsf{F}_s \mid \mathsf{F}_s : M \to G'\}_{s \in S}$ be a (possibly randomised) weak PRF family and let $\varphi : \{-mn, \ldots, mn\} \to G$ be a mapping. Then the following PSA scheme $\Sigma = (\mathsf{Setup}, \mathsf{PSAEnc}, \mathsf{PSADec})$ achieves AO2:*

- **Setup**: $(\mathsf{pp}, T, s_0, s_1, \ldots, s_n) \leftarrow \mathsf{Setup}(1^\kappa)$, *where* pp *are parameters of $G, G', S, M, \mathcal{F}, \varphi$. The keys are $s_i \leftarrow \mathcal{U}(S)$ for all $i \in [n]$ with $s_0 = (*_{i=1}^n s_i)^{-1}$ and $T \subset M$ such that all $t_j \in T$ are chosen uniformly at random from M, $j = 1, \ldots, \lambda = \mathsf{poly}(\kappa)$.*
- **PSAEnc**: *Compute $c_{i,j} = \mathsf{F}_{s_i}(t_j) \cdot \varphi(x_{i,j})$ in G for $x_{i,j} \in \widehat{\mathcal{D}} = \{-m, \ldots, m\}$ and public parameter $t_j \in T$.*
- **PSADec**: *Compute $V_j = \varphi^{-1}(S_j)$ (if possible) with $S_j = \mathsf{F}_{s_0}(t_j) \cdot c_{1,j} \cdot \ldots \cdot c_{n,j}$.*

Moreover, if $\mathcal{F}$ contains only deterministic functions that are homomorphic over S, if φ is homomorphic and injective over $\{-mn, \ldots, mn\}$ and if the $c_{i,j}$ are encryptions of the $x_{i,j}$, then $V_j = \sum_{i=1}^n x_{i,j}$, i.e. then PSADec *correctly decrypts $\sum_{i=1}^n x_{i,j}$.*

The reason for not including the correctness property in the main statement is that in Section 4.2, we will provide an example of a secure PSA scheme based on the DLWE problem that does not have a fully correct decryption algorithm, but a noisy one. This noise is necessary for establishing the security of the protocol and will be also used for preserving the differential privacy of the decryption output.
Hence, 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 $m \cdot n$. Importantly, the product of all pseudo-random values $\mathsf{F}_{s_0}(t_j), \mathsf{F}_{s_1}(t_j), \ldots, \mathsf{F}_{s_n}(t_j)$ is the neutral element in the group G for all $t_j \in T$. Note that the secret keys can be pre-generated using a secure multi-party protocol and hence, no trusted party is required. 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 Theorem 3 does not require a random oracle.

3.1 Security proof

Let **game 1** be the $\mathsf{AO2}$ game from Definition 3 instantiated for the PSA scheme of Theorem 3. We need to show that the advantage $\nu_1(\kappa)$ of a ppt adversary $\mathcal{T}_1$ in winning this game is negligible in the security parameter κ. We define the following intermediate **game 2** for a ppt adversary $\mathcal{T}_2$ 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 κ and returns public parameters pp, time-steps T and secret keys $s_0, s_1, \ldots, s_n$ with $s_0 = (\divideontimes_{i=1}^{n} s_i)^{-1}$. It sends $\kappa, \mathsf{pp}, T, s_0$ to $\mathcal{T}_2$. The challenger flips a random bit $b \leftarrow_R \{0,1\}$. $\mathcal{T}_2$ chooses $U = \{i_1, \ldots, i_u\} \subseteq [n]$ and sends it to the challenger which returns $(s_i)_{i \in [n] \setminus U}$.

Queries. $\mathcal{T}_2$ is allowed to query $(i, t_j, x_{i,j})$ with $i \in U, t_j \in T, x_{i,j} \in \widehat{\mathcal{D}}$ and the challenger returns the following: if $b = 0$, it sends $\mathsf{F}_{s_i}(t_j) \cdot \varphi(x_{i,j})$ to $\mathcal{T}_2$; if $b = 1$, it chooses

$$h_{1,j}, \ldots, h_{u-1,j} \leftarrow \mathcal{U}(G'),\ h_{u,j} := \prod_{i'=1}^{u} \mathsf{F}_{s_{i_{i'}}}(t_j) \cdot \left(\prod_{i'=1}^{u-1} h_{i',j} \right)^{-1}$$

and sends $h_{i,j} \cdot \varphi(x_{i,j})$ to $\mathcal{T}_2$.

Challenge. $\mathcal{T}_2$ chooses $t_{j*} \in T$ such that no encryption query at t_{j*} was made and queries a tuple $(x_{i,j*})_{i \in U}$. If $b = 0$, the challenger sends $(\mathsf{F}_{s_i}(t_{j*}) \cdot \varphi(x_{i,j*}))_{i \in U}$ to $\mathcal{T}_2$; if $b = 1$, it chooses

$$h_{1,j*}, \ldots, h_{u-1,j*} \leftarrow \mathcal{U}(G'),\ h_{u,j*} := \prod_{i'=1}^{u} \mathsf{F}_{s_{i_{i'}}}(t_{j*}) \cdot \left(\prod_{i'=1}^{u-1} h_{i',j*} \right)^{-1}$$

and sends $(h_{i,j*} \cdot \varphi(x_{i,j*}))_{i \in U}$ to $\mathcal{T}_2$.

Queries. $\mathcal{T}_2$ is allowed to make the same type of queries as before with the restriction that no encryption query at t_{j*} can be made.

Guess. $\mathcal{T}_2$ outputs a guess about b.

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

Lemma 4 *For a security parameter κ, let $\mathcal{T}_1$ be an adversary in* **game 1** *with advantage $\nu_1(\kappa) > \mathsf{neg}(\kappa)$. Then there exists an adversary $\mathcal{T}_2$ in* **game 2** *with advantage $\nu_2(\kappa) > \mathsf{neg}(\kappa)$.*

Proof. Given a successful adversary $\mathcal{T}_1$ in **game 1** we construct a successful adversary $\mathcal{T}_2$ in **game 2** as follows:

Setup. Receive $\kappa, \mathsf{pp}, T, s_0$ from the **game 2**-challenger and send it to $\mathcal{T}_1$. Flip a random bit $b' \leftarrow_R \{0,1\}$. Receive $U = \{i_1, \ldots, i_u\} \subseteq [n]$ from $\mathcal{T}_1$ and send it to the challenger. Forward the obtained response $(s_i)_{i \in [n] \setminus U}$ to $\mathcal{T}_1$.

Queries. Forward $\mathcal{T}_1$'s queries $(i, t_j, x_{i,j})$ with $i \in U, t_j \in T, x_{i,j} \in \widehat{\mathcal{D}}$ to the challenger and forward the obtained response $c_{i,j}$ to $\mathcal{T}_1$.

Challenge. $\mathcal{T}_1$ chooses $t_{j*} \in T$ such that no encryption query at t_{j*} was made and queries two different tuples $(x_{i,j*}^{[0]})_{i \in U}, (x_{i,j*}^{[1]})_{i \in U}$ with $\sum_{i \in U} x_{i,j*}^{[0]} = \sum_{i \in U} x_{i,j*}^{[1]}$. Query $(x_{i,j*}^{[b']})_{i \in U}$ to the challenger. Receive back $(c_{i,j*})_{i \in U}$ and forward it to $\mathcal{T}_1$.

Queries. $\mathcal{T}_1$ can make the same type of queries as before with the restriction that no encryption query at t_{j*} can be made.

Guess. $\mathcal{T}_1$ gives a guess about b'. If it is correct, then output 0; if not, output 1.

If $\mathcal{T}_1$ has output the correct guess about b', then $\mathcal{T}_2$ can say with high confidence that the challenge ciphertexts were generated using a weak PRF and therefore outputs 0. On the other hand, if $\mathcal{T}_1$'s guess was not correct, then $\mathcal{T}_2$ can say with high confidence that the challenge ciphertexts were generated using random values and it outputs 1.

Case 1. Let $(c_{i,j*})_{i \in U} = (\mathsf{F}_{s_i}(t_{j*}) \cdot \varphi(x_{i,j*}^{[b']}))_{i \in U}$. Then also the queries were answered using pseudo-random values and thus, $\mathcal{T}_2$ perfectly simulates **game 1** for $\mathcal{T}_1$ and the distribution of the ciphertexts is the same as in **game 1**:

$$\begin{aligned}\Pr[\mathcal{T}_2 \text{ outputs } 0] &= \frac{1}{2}(\Pr[\mathcal{T}_1 \text{ outputs } 0 \,|\, b' = 0] + \Pr[\mathcal{T}_1 \text{ outputs } 1 \,|\, b' = 1]) \\ &= \Pr[\mathcal{T}_1 \text{ wins } \textbf{game 1}] \\ &= \frac{1}{2} + \nu_1(\kappa).\end{aligned}$$

Case 2. Let $(c_{i,j*})_{i \in U} = (h_{i,j*} \cdot \varphi(x_{i,j*}^{[b']}))_{i \in U}$. Then also the queries were answered using random values. The ciphertexts are random with $\prod_{i \in U} c_{i,j*} = \prod_{i \in U} \mathsf{F}_{s_i}(t_{j*}) \cdot \varphi(x_{i,j*}^{[b']})$ such that decryption yields the same sum as in case 1. Because of the perfect security of the one-time pad, the probability that $\mathcal{T}_1$ wins **game 1** is $1/2$ and

$$\begin{aligned}\Pr[\mathcal{T}_2 \text{ outputs } 1] &= \frac{1}{2}(\Pr[\mathcal{T}_1 \text{ outputs } 1 \,|\, b' = 0] + \Pr[\mathcal{T}_1 \text{ outputs } 0 \,|\, b' = 1]) \\ &= \Pr[\mathcal{T}_1 \text{ loses } \textbf{game 1}] \\ &= \frac{1}{2}.\end{aligned}$$

Thus, the advantage of $\mathcal{T}_2$ in winning **game 2** is $\nu_2(\kappa) = \frac{1}{2}\nu_1(\kappa) > \mathsf{neg}(\kappa)$. □

For a ppt adversary $\mathcal{T}_3$, 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 $(i, t_j, x_{i,j})$ the adversary $\mathcal{T}_3$ now just queries (i, t_j) and the challenger in **game 3** sends $\mathsf{F}_{s_i}(t_j)$, if $b = 0$ and $h_{i,j} \leftarrow \mathcal{U}(G')$, if $b = 1$ to the adversary $\mathcal{T}_3$. The rest remains the same as in **game 2**.
It follows immediately that if there exists a successful adversary in **game 2**, then there is also a successful adversary in **game 3**.

Lemma 5 *For a security parameter κ, let $\mathcal{T}_2$ be an adversary in* **game 2** *with advantage $\nu_2(\kappa) > \mathsf{neg}(\kappa)$. Then there exists an adversary $\mathcal{T}_3$ in* **game 3** *with advantage $\nu_3(\kappa) > \mathsf{neg}(\kappa)$.*

Remark 1 *For comparison to the proof of* AO1 *in* [25]*, we emphasise that in the reduction from* AO1 *to an intermediate problem (Proof of Theorem* 1 *in their work), an adversary $\mathcal{B}$ has to compute the ciphertexts $c_{i,j} = g^{x_{i,j}} H(t_j)^{s_i}$ for all users $i \in [n]$ and for all (!) time-steps t_j, since $\mathcal{B}$ does not know in advance for which $i \in [n]$ it will have to use the PRF $H(t_j)^{s_i}$ and for which $i \in [n]$ it will have to use random values. Thus, $\mathcal{B}$ has to program the random oracle H in order to know for all t_j the corresponding random number z_j with $H(t_j) = g^{z_j}$ (where g is a generator) for simulating the original* AO1 *game. In contrast, in our reduction for* AO2*, it is not necessary to program such an oracle, since the simulating adversary $\mathcal{T}_2$ knows in advance the set of uncorrupted users and, for all (!) t_j, it can already decide for which $i \in [n]$ it will use the PRF (which in our case is $t_j^{s_i}$ instead of $H(t_j)^{s_i}$) and for which $i \in [n]$ it will use a random value.*

In the next step, the problem of distinguishing the weak PRF family $\mathcal{F} = \{\mathsf{F}_s : M \to G'\}_{s \in S}$ from a random function family has to be reduced to the problem of winning **game 3**. We use a hybrid argument.

Lemma 6 *For a security parameter κ, let $\mathcal{T}_3$ be an adversary in* **game 3** *with advantage $\nu_3(\kappa)$. Then $\nu_3(\kappa) \leqslant \mathsf{neg}(\kappa)$, if $\mathcal{F}$ is a weak PRF family.*

Proof. We define the following sequence of hybrid games, **game 3**$_l$ with $l = 1, \ldots, u-1$, for a ppt adversary $\mathcal{T}_3$.

Setup. As in **game 2** and **game 3**.
Queries. $\mathcal{T}_3$ is allowed to query multiple (i, t_j) with $i \in U, t_j \in T$ and the challenger returns the following: if $i \notin \{i_1, \ldots, i_{l+b}\}$, it sends $\mathsf{F}_{s_i}(t_j)$ to $\mathcal{T}_3$; if $i \in \{i_1, \ldots, i_{l+b}\}$, it chooses

$$h_{1,j}, \ldots, h_{l-(1-b),j} \leftarrow \mathcal{U}(G'),\ h_{l+b,j} := \prod_{i'=1}^{l+b} \mathsf{F}_{s_{i_{i'}}}(t_j) \cdot \left(\prod_{i'=1}^{l-(1-b)} h_{i',j} \right)^{-1}$$

and sends the $h_{i,j}$ to $\mathcal{T}_3$.

Challenge. $\mathcal{T}_3$ chooses $t_{j*} \in T$ such that no encryption query at t_{j*} was made. The challenger chooses

$$h_{1,j*}, \ldots, h_{l-(1-b),j*} \leftarrow \mathcal{U}(G'),\ h_{l+b,j*} := \prod_{i'=1}^{l+b} \mathsf{F}_{s_{i_{i'}}}(t_{j*}) \cdot \left(\prod_{i'=1}^{l-(1-b)} h_{i',j*} \right)^{-1}$$

and sends the following sequence to $\mathcal{T}_3$:

$$(h_{1,j*}, \ldots, h_{l+b,j*}, \mathsf{F}_{s_{i_{l+b+1}}}(t_{j*}), \ldots, \mathsf{F}_{s_{i_u}}(t_{j*})).$$

Queries. $\mathcal{T}_3$ can make the same type of queries as before with the restriction that no encryption query at t_{j*} can be made.

Guess. $\mathcal{T}_3$ outputs a guess about b.

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

It is immediate that **game 3**$_1$ with $b = 0$ corresponds to the case $b = 0$ in **game 3** and **game 3**$_{u-1}$ with $b = 1$ corresponds to the case $b = 1$ in **game 3**. Moreover the ciphertexts in **game 3**$_l$ with $b = 1$ have the same distribution as the ciphertexts in **game 3**$_{l+1}$ with $b = 0$. Therefore

$$\Pr[\mathcal{T}_3 \text{ wins } \textbf{game } \mathbf{3}_{l+1} \,|\, b = 0] = \Pr[\mathcal{T}_3 \text{ loses } \textbf{game } \mathbf{3}_l \,|\, b = 1].$$

Using an adversary $\mathcal{T}_3$ in **game 3**$_l$ we construct an efficient ppt distinguisher $\mathcal{D}_{\mathrm{PRF}}$ which has access to an oracle $\mathcal{O}(\cdot) \leftarrow_R \{\mathsf{F}_{s'}(\cdot), \mathsf{rand}(\cdot)\}$, where $s' \leftarrow \mathcal{U}(S)$, $\mathsf{F}_{s'} : M \to G'$ is a weak PRF and $\mathsf{rand} : M \to G'$ is a random function. $\mathcal{D}_{\mathrm{PRF}}$ gets κ as input and proceeds as follows.

1. Choose $s_0 \leftarrow \mathcal{U}(S)$, generate pp and T with $t_j \leftarrow \mathcal{U}(M)$ for all $t_j \in T$. Compute $\mathsf{F}_{s_0}(t_j)$ for all $t_j \in T$.
2. Make oracle queries for t_j and receive $\mathcal{O}(t_j)$ for all $t_j \in T$.
3. Send $\kappa, \mathsf{pp}, T, s_0$ to $\mathcal{T}_3$.
4. Receive $U = \{i_1, \ldots, i_u\} \subseteq [n]$ from $\mathcal{T}_3$. For all $i \in [n] \backslash \{i_l, i_{l+1}\}$ choose $s_i \leftarrow \mathcal{U}(S)$. Send $(s_i)_{i \in [n] \backslash U}$ to $\mathcal{T}_3$.
5. **Queries.** If $\mathcal{T}_3$ queries (i, t_j) with $i \in U, t_j \in T$, then return the following: if $i \notin \{i_1, \ldots, i_{l+1}\}$, send $\mathsf{F}_{s_i}(t_j)$ to $\mathcal{T}_3$; if $i = i_{l+1}$, send $\mathcal{O}(t_j)$ to $\mathcal{T}_3$; if $i \in \{i_1, \ldots, i_l\}$, choose $h_{1,j}, \ldots, h_{l-1,j} \leftarrow \mathcal{U}(G')$ and
$$h_{l,j} := \left(\mathsf{F}_{s_0}(t_j) \cdot \mathcal{O}(t_j) \cdot \prod_{i'=1}^{l-1} h_{i',j} \cdot \prod_{i \in [n] \backslash \{i_1, \ldots, i_{l+1}\}} \mathsf{F}_{s_i}(t_j) \right)^{-1}$$
and send the $h_{i,j}$ to $\mathcal{T}_3$.
6. **Challenge.** $\mathcal{T}_3$ chooses $t_{j*} \in T$ such that no encryption query at t_{j*} was made. Choose $h_{1,j*}, \ldots, h_{l-1,j*} \leftarrow \mathcal{U}(G')$ and
$$h_{l,j*} := \left(\mathsf{F}_{s_0}(t_{j*}) \cdot \mathcal{O}(t_{j*}) \cdot \prod_{i'=1}^{l-1} h_{i',j*} \cdot \prod_{i \in [n] \backslash \{i_1, \ldots, i_{l+1}\}} \mathsf{F}_{s_i}(t_{j*}) \right)^{-1}$$

and send the sequence $(h_{1,j*}, \ldots, h_{l,j*}, \mathcal{O}(t_{j*}), \mathsf{F}_{s_{i_{l+2}}}(t_{j*}), \ldots, \mathsf{F}_{s_{i_u}}(t_{j*}))$ to $\mathcal{T}_3$.

7. **Queries.** $\mathcal{T}_3$ can make the same type of queries as before with the restriction that no encryption query at t_{j*} can be made.
8. **Guess.** $\mathcal{T}_3$ outputs a guess about whether the $l+1^{\text{th}}$ element is random or pseudo-random. Output the same guess.

If $\mathcal{T}_3$ has output the correct guess about whether the $l+1^{\text{th}}$ element is random or pseudo-random, then $\mathcal{D}_{\text{PRF}}$ can distinguish between $\mathsf{F}_{s'}(\cdot)$ and $\mathsf{rand}(\cdot)$. Now we prove this result formally and show that **game 3**$_l$ is perfectly simulated by $\mathcal{T}_3$.

Case 1. Let $\mathcal{O}(\cdot) = \mathsf{F}_{s'}(\cdot)$. Define $s_{i_{l+1}} := s'$. Since S is a group, there exists an element s_{i_l} with $s_{i_l} = (s_0 * \divideontimes_{i \in [n] \setminus \{i_l\}} s_i)^{-1}$ and for all $t_j \in T$:

$$\left(\mathsf{F}_{s_0}(t_j) \cdot \mathsf{F}_{s'}(t_j) \cdot \prod_{i \in [n] \setminus \{i_1, \ldots, i_{l+1}\}} \mathsf{F}_{s_i}(t_j) \right)^{-1} = \prod_{i'=1}^{l} \mathsf{F}_{s_{i_{i'}}}(t_j).$$

Then for all $t_j \in T$, the value $h_{l,j}$ is equal to

$$\left(\mathsf{F}_{s_0}(t_j) \cdot \prod_{i'=1}^{l-1} h_{i',j} \cdot \mathsf{F}_{s'}(t_j) \cdot \prod_{i \in [n] \setminus \{i_1, \ldots, i_{l+1}\}} \mathsf{F}_{s_i}(t_j) \right)^{-1} = \prod_{i'=1}^{l} \mathsf{F}_{s_{i_{i'}}}(t_j) \cdot \left(\prod_{i'=1}^{l-1} h_{i',j} \right)^{-1}.$$

The distribution of the ciphertexts corresponds to the case in **game 3**$_l$ with $b = 0$.

Case 2. Let $\mathcal{O}(\cdot) = \mathsf{rand}(\cdot)$. Define the random elements $h_{l+1,j} := \mathsf{rand}(t_j)$ for all $t_j \in T$. Since S, M are groups, there exists an element $s' \in S$ with $s' = (s_0 * \divideontimes_{i \in [n] \setminus \{i_l, i_{l+1}\}} s_i)^{-1}$. Let $s_{i_l} \leftarrow \mathcal{U}(S)$ and $s_{i_{l+1}} := s' * s_{i_l}^{-1}$. Then for all $t_j \in T$:

$$\left(\mathsf{F}_{s_0}(t_j) \cdot \prod_{i \in [n] \setminus \{i_1, \ldots, i_{l+1}\}} \mathsf{F}_{s_i}(t_j) \right)^{-1} = \prod_{i'=1}^{l+1} \mathsf{F}_{s_{i_{i'}}}(t_j)$$

and the value $h_{l,j}$ is equal to

$$\left(\mathsf{F}_{s_0}(t_j) \cdot h_{l+1,j} \cdot \prod_{i'=1}^{l-1} h_{i',j} \cdot \prod_{i \in [n] \setminus \{i_1, \ldots, i_{l+1}\}} \mathsf{F}_{s_i}(t_j) \right)^{-1}$$

and equivalently

$$h_{l+1,j} = \prod_{i'=1}^{l+1} \mathsf{F}_{s_{i_{i'}}}(t_j) \cdot \left(\prod_{i'=1}^{l} h_{i',j} \right)^{-1}.$$

The distribution of the ciphertexts corresponds to the case in **game 3**$_l$ with $b = 1$.

Without loss of generality, let

$$\Pr[\mathcal{T}_3 \text{ wins } \textbf{game } \mathbf{3}_l \mid b = 0] \geqslant \Pr[\mathcal{T}_3 \text{ loses } \textbf{game } \mathbf{3}_l \mid b = 1].$$

In total we obtain

$$\begin{aligned}
&\Pr[\mathcal{T}_3 \text{ wins } \textbf{game } \mathbf{3}_l \mid b = 0] - \Pr[\mathcal{T}_3 \text{ loses } \textbf{game } \mathbf{3}_l \mid b = 1] \\
&= \Pr[\mathcal{D}_{\mathrm{PRF}}^{\mathsf{F}_{s'}(\cdot)}(\kappa) = 1] - \Pr[\mathcal{D}_{\mathrm{PRF}}^{\mathsf{rand}(\cdot)}(\kappa) \ = 1] \\
&\leqslant \Pr[\mathcal{D}_{\mathrm{PRF}}^{\mathsf{F}_{s'}(\cdot)}(\kappa) = 1] - \Pr[\mathcal{D}_{\mathrm{PRF}}^{\mathsf{rand}(\cdot)}(\kappa) = 1].
\end{aligned}$$

This expression is negligible by the pseudo-randomness of $\mathsf{F}_{s'}(\cdot)$ on uniformly chosen input. Therefore, the advantage of $\mathcal{T}_3$ in winning $\textbf{game } \mathbf{3}_l$ is negligible. Finally, by a hybrid argument we have:

$$\begin{aligned}
&\Pr[\mathcal{T}_3 \text{ wins } \textbf{game } \mathbf{3}] \\
&= \frac{1}{2}(\Pr[\mathcal{T}_3 \text{ wins } \textbf{game } \mathbf{3} \mid b = 0] + \Pr[\mathcal{T}_3 \text{ wins } \textbf{game } \mathbf{3} \mid b = 1]) \\
&= \frac{1}{2}(\Pr[\mathcal{T}_3 \text{ wins } \textbf{game } \mathbf{3}_1 \mid b = 0] + \Pr[\mathcal{T}_3 \text{ wins } \textbf{game } \mathbf{3}_{u-1} \mid b = 1]) \\
&= \frac{1}{2} + \frac{1}{2}(\Pr[\mathcal{T}_3 \text{ wins } \textbf{game } \mathbf{3}_1 \mid b = 0] - \Pr[\mathcal{T}_3 \text{ loses } \textbf{game } \mathbf{3}_{u-1} \mid b = 1]) \\
&= \frac{1}{2} + \frac{1}{2}\sum_{l=1}^{u-1} \Pr[\mathcal{T}_3 \text{ wins } \textbf{game } \mathbf{3}_l \mid b = 0] - \Pr[\mathcal{T}_3 \text{ loses } \textbf{game } \mathbf{3}_l \mid b = 1] \\
&= \frac{1}{2} + (u-1) \cdot \mathsf{neg}(\kappa).
\end{aligned}$$

□

We can now complete the proof of Theorem 3.

Proof (Proof of Theorem 3). By Lemma 4 - 6:
$\nu_1(\kappa) = 2 \cdot \nu_2(\kappa) = 2 \cdot \nu_3(\kappa) = 2 \cdot (u-1) \cdot \mathsf{neg}(\kappa) < 2 \cdot n \cdot \mathsf{neg}(\kappa) = \mathsf{neg}(\kappa)$. □

4 The Constructions

In this section, we provide two different PSA schemes that use a key-homomorphic weak PRF. The first construction is based on the DDH assumption and is a modification of the scheme from [25]. We will compare the practical performances of these schemes. The second construction is based on the DLWE assumption and incorporates a differentially private mechanism with discrete Gaussian noise.

4.1 A DDH-based PSA scheme

We provide an instantiation of a secure PSA scheme consisting of efficient algorithms. It is constructed from a DDH-based key-homomorphic weak PRF. Hence, its security is based on the DDH assumption making it comparable to the scheme from [25].

Example 2 *Let $q > m \cdot n$ and $p = 2 \cdot q + 1$ be large primes. Let furthermore $G = \mathbb{Z}_{p^2}^*, S = \mathbb{Z}_{pq}, M = G' = \mathcal{QR}_{p^2}$ and $g \in \mathbb{Z}_{p^2}^*$ with $ord(g) = pq$. Then g generates the group $M = G' = \mathcal{QR}_{p^2}$ of quadratic residues modulo p^2. In this group, we make the assumption that the DDH problem is hard.*[2] *Then we define*

- *Let $\mathsf{pp} = (g, p)$. Choose keys $s_1, \ldots, s_n \leftarrow \mathcal{U}(\mathbb{Z}_{pq})$ and $s_0 \equiv -\sum_{i=1}^{n} s_i \bmod pq$. Let $T \subset M$ with $|T| = \lambda$, i.e. t_j is a power of g for every $t_j \in T$, $j = 1, \ldots, \lambda$.*
- *$\mathsf{F}_{s_i}(t_j) \equiv t_j^{s_i} \bmod p^2$. This is a weak PRF under the DDH assumption in $\mathcal{QR}_{p^2}$ (which can be shown using similar arguments as in [20]).*
- *$\varphi(x_{i_j}) \equiv 1 + p \cdot x_{i,j} \bmod p^2$, where $-m \leqslant x_{i,j} \leqslant m$. (It is immediate that φ is homomorphic and injective over $\{-mn, \ldots, mn\}$.)*

For decryption and aggregation, compute $V_j \in \{1 - p \cdot mn, \ldots, 1 + p \cdot mn\}$ with

$$\begin{aligned} V_j &\equiv \mathsf{F}_{s_0}(t_j) \cdot \prod_{i=1}^{n} \mathsf{F}_{s_i}(t_j) \cdot \varphi(x_{i,j}) \equiv \prod_{i=1}^{n} (1 + p \cdot x_{i,j}) \\ &\equiv 1 + p \cdot \sum_{i=1}^{n} x_{i,j} + p^2 \cdot \sum_{i,i' \in [n], i' \neq i} x_{i,j} x_{i',j} + \ldots + p^n \cdot \prod_{i=1}^{n} x_{i,j} \\ &\equiv 1 + p \cdot \sum_{i=1}^{n} x_{i,j} \bmod p^2 \end{aligned}$$

and decrypt $\sum_{i=1}^{n} x_{i,j} = \frac{1}{p}(V_j - 1)$ over the integers.

Remark 2 *In the random oracle model, the construction shown in Example 2 achieves the stronger notion of $\mathsf{AO1}$. For details, see the proof in Section A of the appendix in [25]. The same proof can be applied to our instantiation by simply replacing the map φ involved and using a strong version of the PRF F.*

Differential Privacy. In connection with a differentially private mechanism, a PSA scheme assures that the analyst is only able to learn a noisy aggregate of users' data (as close as possible to the real answer) and nothing else. More specifically, for preserving differential privacy, it would be sufficient to add a single copy of (properly distributed) noise Y to the aggregated statistics. Since we deal with a distributed setting, the noise cannot be added, once the aggregate has been computed. Hence, the users themselves have to generate and add noise to their original data in such a way that the sum of the errors has the same distribution as Y. For this purpose, we see two different approaches. In the first one, with small probability, a user adds noise sufficient to preserve the privacy

[2] This is reasonable because the simple Legendre attack, i.e. decide whether $w = g^{xy}$ or $w = g^z$ given a generator g and g^x, g^y, w by computing and comparing the Legendre symbols of g^x, g^y, w over p^2, does not work here, since in this group there is always some h with $g \equiv h^2 \bmod p^2$. Thus, the Legendre symbol of g^v over p^2 is 1 *for all* v and this attack does not provide any way to distinguish g^{xy} from g^z.

Table 1: Time measurements of the schemes for different parameters

(a) Encryption

Length of p	1024-bit	2048-bit	4096-bit
[25]	1.1 ms	7.5 ms	57.0 ms
This work	3.9 ms	29.4 ms	225.0 ms

(b) Decryption (2048 bit, $n = 1000$)

m	10^1	10^2	10^3
[25], brute-force	0.24 s	2.67 s	28.97 s
This work	0.08 s	0.08 s	0.09 s

of the entire statistics. This probability is calibrated in such a way only one of the n users is actually expected to add noise at all. Shi et al. [25] investigate this method using the geometric mechanism by Ghosh et al. [11]. In the second approach, each user generates noise of small variance, such that the sum of all noisy terms suffices to preserve differential privacy of the aggregate.[3] To achieve this goal, we need a *discrete* probability distribution which is *closed under convolution* and is known to provide differential privacy. The binomial mechanism by Dwork et al. [9] and the discrete Gaussian mechanism introduced in the next subsection serve these purposes.[4] Note that due to the use of a computationally secure protocol, we achieve differential privacy also only against ppt adversaries. For this case, the notion of computational differential privacy was introuced in [19]. We can prove a composition theorem showing that the use of a differentially private mechanism within a secure PSA scheme preserves computational differential privacy. This is left for a follow-up work.

Comparison of DDH-based schemes. Whereas the PRF in Example 1 is similar to the one used in Example 2 (the underlying group G is $\mathbb{Z}_p^*$ rather than $\mathbb{Z}_{p^2}^*$), the aggregational function is defined by $\varphi(x_{i,j}) = g^{x_{i,j}} \bmod p$, 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. 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 super-polynomially large.

We compare the practical running times for encryption and decryption of the scheme from [25] with the algorithms of our scheme in Table 1a and Table 1b, 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 are executed on an Intel Core i5, 64-bit CPU at 2.67 GHz. We compare the schemes at the same security level, assuming that the DDH problem modulo p is as hard as modulo p^2, i.e. we use the same value for p in both schemes.

[3] One can assume that corrupted users will not add any noise to their data in order to help the analyst to compromise the privacy of the remaining users. For simplicity, we ignore this issue here.

[4] 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 noise.

For different bit-lengths of p, we observe that the encryption of our scheme is roughly 4 times slower than the encryption of the scheme from [25]. The running time of our decryption algorithm is widely dominated by the aggregation phase. Therefore it is clear, that it linearly depends on n. The same holds for the brute-force decryption of the scheme in [25], since the range for the discrete logarithm also linearly grows with n. Using a 2048-bit prime and fixing $n = 1000$, the running time of the decryption in our scheme is less than 0.1 seconds for varying values of m. In contrast, the time for brute-force decryption in [25] grows roughly linearly in m.

4.2 A DLWE-based PSA scheme

We construct a secure PSA scheme from a weak PRF construction based on Theorem 2 that automatically preserves DP. We analyse the privacy and accuracy guarantees of this scheme and also the trade-off between security and accuracy.

Example 3 *We can build an instantiation of Theorem* 3 *(without correct decryption) based on the* DLWE *problem as follows. Set* $S = M = \mathbb{Z}_q^\kappa, G = \mathbb{Z}_q$*, choose* $\mathbf{s}_i \leftarrow \mathcal{U}(\mathbb{Z}_q^\kappa)$ *for all* $i = 1, \ldots, n$ *and* $\mathbf{s}_0 = -\sum_{i=1}^n \mathbf{s}_i$*, set* $\mathsf{F}_{\mathbf{s}_i}(\mathbf{t}_j) = \langle \mathbf{t}_j, \mathbf{s}_i \rangle + e_{i,j}$*, such that* $e_{i,j} \leftarrow D(\nu/n)$ *with parameter* $\nu/n = 2\kappa/\pi$ *(by Theorems* 1 *and* 2*, this is a so-called randomised weak pseudo-random function as described in* [1] *and in* [2]*), and let* φ *be the identity function. Therefore* $\langle \mathbf{t}_j, \mathbf{s}_i \rangle + e_{i,j} + d_{i,j} = c_{i,j} \leftarrow \mathsf{PSAEnc}_{\mathbf{s}_i}(\mathbf{t}_j, d_{i,j})$ *for data value* $d_{i,j} \in \mathbb{Z}_q$*,* $i = 1, \ldots, n$*. The decryption function is defined by*

$$\mathsf{PSADec}_{\mathbf{s}_0}(\mathbf{t}_j, c_{1,j}, \ldots, c_{n,j}) = \langle \mathbf{t}_j, \mathbf{s}_0 \rangle + \sum_{i=1}^n c_{i,j} = \langle \mathbf{t}_j, \mathbf{s}_0 \rangle + \sum_{i=1}^n \mathsf{F}_{\mathbf{s}_i}(\mathbf{t}_j) + d_{i,j}$$
$$= \sum_{i=1}^n d_{i,j} + \sum_{i=1}^n e_{i,j}.$$

Thus, the decryption is not perfectly correct anymore, but yields a noisy aggregate. The associated DLWE *problem is hard and the above scheme is secure.*

Remark 3 *The original result by Regev* [23] *states that the LWE problem is hard in the set* $\mathbb{T} = \mathbb{R}/\mathbb{Z}$ *when the noise is distributed according to the* continuous *Gaussian distribution modulo* 1*. Although the continuous Gaussian distribution is reproducible as well, it does not seem to fit well for a DLWE-based PSA scheme. For data processing reasons, the values would have to be discretised. Therefore the resulting noise would follow a distribution which is not reproducible in general. However, in* [5] *it was shown that the sum of* n *discrete Gaussians each with parameter* ν *is statistically close to a discrete Gaussian with parameter* $n\nu$*, if* $\nu > n\eta_{\epsilon'}(\mathcal{L})^2$ *for some smoothing parameter* $\eta_{\epsilon'}(\mathcal{L})$ *of the underlying lattice* $\mathcal{L}$*.*

Differential Privacy. We show that for the final aggregate $\sum_{i=1}^{n} d_{i,j} + e_{i,j}$ in Example 3, (ϵ, δ)-DP is preserved. First, assume due to Remark 3 that $\sum_{i=1}^{n} e_{i,j}$ is distributed according to $D(\nu)$ if the $e_{i,j}$ are i.i.d. according to $D(\nu/n)$.

Theorem 7 (Discrete Gaussian Mechanism) *Let $0 < \epsilon, \delta < 1$. For all databases $D \in \mathcal{D}^n$, the randomised mechanism $\mathcal{A}(D) := f(D) + Y$ preserves (ϵ, δ)-DP with respect to any query f with sensitivity $S(f)$, if Y is distributed according $D(\nu)$ with $\nu = 4\pi S(f)^2 \log(2/\delta)/\epsilon^2$.*

Proof. Let $D_0, D_1 \in \mathcal{D}^n$ be adjacent databases with $|f(D_0) - f(D_1)| \leqslant S(f)$. The largest ratio between $\Pr[\mathcal{A}(D_0) = R]$ and $\Pr[\mathcal{A}(D_1) = R]$ is reached when $k := R - f(D_1) - S(f) = R - f(D_0) \geqslant 0$, where R is any possible output of $\mathcal{A}$. Then for all possible outputs R of $\mathcal{A}$:

$$\begin{aligned}\frac{\Pr[\mathcal{A}(D_0) = R]}{\Pr[\mathcal{A}(D_1) = R]} &= \frac{\Pr[Y = k]}{\Pr[Y = k + S(f)]} = \frac{\exp(-\pi k^2/\nu)}{\exp(-\pi(k + S(f))^2/\nu)} \\ &= \exp((\pi/\nu)((k + S(f))^2 - k^2)) = \exp((\pi/\nu)(2kS(f) + S(f)^2)) \\ &\leqslant \exp(\epsilon)\end{aligned}$$

if $k \leqslant (\epsilon\nu)/(2\pi S(f)) - S(f)/2 =: B$. We bound the complementary probability:

$$\begin{aligned}\Pr[Y > B] \leqslant \frac{\sqrt{\nu}}{c_\nu} \cdot \int_B^\infty \frac{1}{\sqrt{\nu}} \cdot \exp(-\pi x^2/\nu)\mathrm{d}x &\leqslant \frac{1}{c_\nu} \cdot \frac{\sqrt{\nu}}{B\sqrt{2\pi}} \cdot \exp(-\pi B^2/\nu) \\ &\leqslant 2\exp(-\epsilon^2\nu/(4\pi S(f)^2)) \qquad (1)\end{aligned}$$

which is $\leqslant \delta$, if $\nu \geqslant 4\pi S(f)^2 \log(2/\delta)/\epsilon^2$. This implies that $\nu > 2\pi S(f)^2/\epsilon^2$ and since $\epsilon < 1$ and $c_\nu > 1$, Inequality (1) follows. □

We compute the (α, β)-accuracy of the discrete Gaussian mechanism as

$$\beta = \Pr[|Y| > \alpha] \leqslant 2\frac{\sqrt{\nu}}{\alpha c_\nu \sqrt{2\pi}} \cdot \exp(-\pi\alpha^2/\nu) \leqslant \frac{\sqrt{\nu}}{c_\nu} \cdot \exp(-\pi\alpha^2/\nu)$$

for $\alpha \geqslant \sqrt{2/\pi}$ and therefore $\alpha \leqslant \sqrt{\nu \log(\sqrt{\nu}/(c_\nu\beta))/\pi} = \tilde{O}(S(f)/\epsilon)$. This bound is similar to standard solutions (e.g. [10], [9]), especially to the binomial mechanism, that would be a standard solution for (ϵ, δ)-DP in the distributed setting. Hence, we have two bounds on ν: one for security and one for (ϵ, δ)-DP. Assume all users generate their noise according to the required security level as in Example 3. We can compute the level of (ϵ, δ)-DP that is thus preserved. Therefore we set the two bounds on ν to be equal and solve for ϵ:

$$2n\kappa/\pi = 4\pi S(f)^2 \log(2/\delta)/\epsilon^2 \Leftrightarrow \epsilon = \epsilon(\kappa) = S(f)\pi\sqrt{2\log(2/\delta)/(n\kappa)}.$$

Thus, in addition to a privacy/accuracy trade-off there is also a security/accuracy trade-off depending on κ and n, since $\alpha = \tilde{O}(S(f)/\epsilon) = \tilde{O}(\sqrt{n\kappa})$.

5 Conclusion

We investigated cryptographic methods for privacy-preserving data analysis in the distributed setting and showed, that a secure PSA scheme for time-series data can be built upon any key-homomorphic weak PRF. Opposed to previous work, our proof works in the standard model and our security reduction is tighter. We provided two instantiations of this result. The first PSA scheme is based on the DDH assumption and has an efficient decryption algorithm even for a super-polynomially large data universe, as opposed to previous work. The other PSA scheme is based on the DLWE assumption with an inherent differentially private mechanism, leading to the first lattice-based secure PSA scheme. An open challenge is to prove that the composition of a PSA scheme with a differentially private mechanism provides computational differential privacy. Another interesting challenge is the following: due to Remark 3, in our DLWE-based scheme, we need $\nu > n\eta_{\epsilon'}(\mathcal{L})^2$ for reproducibility of the error distribution which is in turn necessary for differential privacy. Thus, ν depends on n. It is desirable to construct a lattice-based PSA scheme, where the application of a differentially private mechanism is independent of the number of users.

References

1. Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *Proc. of CRYPTO '09*, pages 595–618, 2009.
2. Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In *Proc. of EUROCRYPT '12*, pages 719–737. 2012.
3. Fabrice Benhamouda, Marc Joye, and Benoît Libert. A new framework for privacy-preserving aggregation of time-series data. *ACM Transactions on Information and System Security*, 18(3), 2016.
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. Dan Boneh and David Mandell Freeman. Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In *Proc. of PKC' 11*, pages 1–16, 2011.
6. Chris Clifton and Tamir Tassa. On syntactic anonymity and differential privacy. In *Proc. of ICDE Workshops '13*, pages 88–93, 2013.
7. Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *Proc. of EUROCRYPT '02*, pages 45–64, 2002.
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, 2006.
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. Arpita Ghosh, Tim Roughgarden, and Mukund Sundararajan. Universally utility-maximizing privacy mechanisms. In *Proc. of STOC '09*, pages 351–360, 2009.
12. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.
13. 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.
14. N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *Proc. of ICDE '07*, pages 106–115, 2007.
15. Yehuda Lindell and Benny Pinkas. Secure multiparty computation for privacy-preserving data mining. *Journal of Privacy and Confidentiality*, 1(1):5, 2009.
16. Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. *Proc. of Knowl. Discov. Data '07*, 2007.
17. Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *Proc. of FOCS '07*, pages 94–103, 2007.
18. Daniele Micciancio and Petros Mol. Pseudorandom knapsacks and the sample complexity of lwe search-to-decision reductions. In *Proc. of CRYPTO '11*, pages 465–484. 2011.
19. Ilya Mironov, Omkant Pandey, Omer Reingold, and Salil Vadhan. Computational differential privacy. In *Proc. of CRYPTO '09*, pages 126–142, 2009.
20. Moni Naor, Benny Pinkas, and Omer Reingold. Distributed pseudo-random functions and KDCs. In *Proc. of EUROCRYPT '99*, pages 327–346, 1999.
21. 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.
22. Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In *Proc. of STOC '09*, pages 333–342, 2009.
23. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proc. of STOC '05*, pages 84–93, 2005.
24. Pierangela Samarati and Latanya Sweeney. Generalizing data to provide anonymity when disclosing information (abstract). In *Proc. of PODS '98*, pages 188–, 1998.
25. 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.