# Decentralized Private Stream Aggregation from Lattices

Uddipana Dowerah[1,2(✉)] and Aikaterini Mitrokotsa[2]

[1] Chalmers University of Technology, Gothenburg, Sweden
[2] University of St. Gallen, St. Gallen, Switzerland
{uddipana.dowerah,katerina.mitrokotsa}@unisg.ch

**Abstract.** As various industries and government agencies increasingly seek to build quantum computers, the development of post-quantum constructions for different primitives becomes crucial. Lattice-based cryptography is one of the top candidates for constructing quantum-resistant primitives. In this paper, we propose a decentralized Private Stream Aggregation (PSA) protocol based on the Learning with Errors (LWE) problem. PSA allows secure aggregation of time-series data over multiple users without compromising the privacy of the individual data. In almost all previous constructions, a trusted entity is used for the generation of keys. We consider a scenario where the users do not want to rely on a trusted authority. We, therefore, propose a decentralized PSA (DPSA) scheme where each user generates their own keys without the need for a trusted setup. We give a concrete construction based on the hardness of the LWE problem both in the random oracle model and in the standard model.

**Keywords:** Private Stream Aggregation · Learning with Errors · Post-quantum cryptography · Decentralized

## 1 Introduction

The growing interest in building quantum computers has led to a widespread need for the development of post-quantum cryptographic protocols. Lattice-based cryptography is among the best candidates for post-quantum cryptography due to its versatility and resistance to quantum attacks. The hardness of lattice-based cryptographic algorithms is based on the assumed worst-case hardness of lattice problems. A well-known computational problem based on lattices is the Learning with Errors (LWE) problem introduced in [19]. In this paper, we focus on constructing a Private Stream Aggregation (PSA) protocol based on the LWE problem.

In various real-world scenarios, a data aggregator may seek to collect data from multiple organizations or individuals to compute various statistics over the data. However, a significant challenge in such applications is to ensure the privacy of the participants, particularly when the aggregator is not trusted. Certain


C. Pöpper and L. Batina (Eds.): ACNS 2024, LNCS 14584, pp. 189–214, 2024.
https://doi.org/10.1007/978-3-031-54773-7_8

examples of such applications include personal identifiable information such as social security numbers, financial data such as credit card details, medical data such as health records, or educational data such as transcripts, etc. This motivated the construction of private stream aggregation protocols that preserves individual data privacy and enables secure aggregation of time-series data across multiple users.

In a PSA protocol, there are multiple clients and one untrusted aggregator. Each client sends an encrypted message over a time period, usually called a timestamp (also called a label in some papers [15]), to the aggregator and the aggregator decrypts the sum of the messages over that time period without the knowledge of the individual messages. Timestamps are used to prevent the aggregator from mixing ciphertexts with different timestamps which in turn prevents the leakage of information about the values of individual clients. The security of a PSA protocol is captured by the notion of *aggregator obliviousness* which requires that the aggregator learns nothing more than the aggregated sum. A PSA protocol remains secure even in situations where the aggregator colludes with a subset of clients. In this case, the aggregator can only learn the sum of the messages from the non-colluding clients. A possible application scenario for PSA is Smart Grids where PSA can be used to collect and analyze real-time energy consumption data from different households or businesses for load balancing, energy management, or renewable energy integration, while maintaining the privacy of the customers. Another possible application is Traffic Management where it can be used to collect and analyze real-time traffic data from different sensors or vehicles for traffic prediction, route optimization, or accident prevention, while preserving the privacy of individuals. Private stream aggregation can also be applied in federated learning to enable the aggregation of locally trained models from multiple devices, while preserving privacy. In federated learning, each device trains a model on its local data and sends the updated model to a centralized server for aggregation. However, the privacy of local data is a major concern in this process.

Furthermore, to provide an additional layer of privacy protection, differential privacy can be used with PSA [21]. Private stream aggregation with differential privacy involves the addition of noise to the data prior to aggregation. The amount of noise added is defined by a privacy budget that limits the amount of information that can be revealed about an individual. Various PSA constructions consider the distributed model of differential privacy, where the clients add differentially private noise to their data [21,24] before encryption. In this paper, we do not explicitly consider differential privacy in our construction. However, similar procedures can be adopted as in previous works [21] to add differentially private noise to the inputs.

A closely related notion to private stream aggregation is Multi-Client Functional Encryption (MCFE) for inner products. In contrast to traditional public key encryption that either decrypts the entire message or nothing, Functional Encryption (FE) allows a user to learn specific functions of the encrypted data without disclosing any other information. More specifically, in FE, a secret key

$\mathsf{sk}_f$ is associated to a function $f$ and the ciphertext $\mathsf{ct}_{\mathbf{x}}$ encrypts a message $\mathbf{x}$ and decrypting $\mathsf{ct}_{\mathbf{x}}$ with $\mathsf{sk}_f$ reveals $f(\mathbf{x})$ and nothing else. In Inner Product Functional Encryption (IPFE), the ciphertext $\mathsf{ct}_{\mathbf{x}}$ is associated to a message vector $\mathbf{x}$ and the secret keys $\mathsf{sk}_{\mathbf{y}}$ can be generated with respect to some vector $\mathbf{y}$, while the decryption of $\mathsf{ct}_{\mathbf{x}}$ with $\mathsf{sk}_{\mathbf{y}}$ recovers the inner product $\langle \mathbf{x}, \mathbf{y} \rangle$. In inner product MCFE, there are multiple clients and one or more aggregators. Each user encrypts their input $x_i$ using a secret key $\mathsf{sk}_i$ and sends the ciphertext $\mathsf{ct}_{x_i}$ to the aggregator. Using the functional key $\mathsf{sk}_{\mathbf{y}}$, the aggregator recovers the inner product $\langle \mathbf{x}, \mathbf{y} \rangle = \sum_i x_i y_i$ where $\mathbf{x} := (x_1, x_2, \ldots)$ and $\mathbf{y} := (y_1, y_2, \ldots)$. Observe that for the all ones vector $\mathbf{y} = (1, \ldots, 1)$, this is exactly a PSA scheme. Therefore, PSA can be seen as a specific case of MCFE for the evaluation of inner products where only a single key corresponding to the vector $\mathbf{y} = (1, \ldots, 1)$ is revealed to the aggregator.

### 1.1 Our Contributions

Almost all known PSA schemes [4,7,13,15,21,23,24,27] use a trusted authority for key generation that generates the client keys for encryption and aggregator key for decryption. However, since the main goal of PSA is to allow an untrusted aggregator to perform aggregate statistics without compromising individual data, the use of a trusted authority is not aligned with the objectives of PSA. The use of a trusted authority can be avoided by decentralizing the setup and key generation procedure.

In this paper, we propose a decentralized private stream aggregation (DPSA) protocol that does not rely on a trusted authority for key generation. We take inspiration from the decentralized multi-client functional encryption scheme proposed in [12]. In the DPSA scheme, the clients generate their own keys and share it with the aggregator in a secure way such that the aggregator does not learn the individual client keys and only learns the aggregator decryption key which is equal to the sum of the client keys. We first give a construction in the random oracle model using a hash function modeled as a random oracle. We then show how to modify it into a construction in the standard model using a weak pseudorandom function (PRF). For the standard model we modify the ideas from [26] to achieve a construction with unbounded timestamps. Our scheme achieves aggregator oblivious security with *static corruptions* based on the LWE problem. If instantiated with a trusted setup, the protocol achieves aggregator obliviousness with *adaptive corruptions*. We also discuss possible solutions for practical deployments such as clients joining and leaving the system. Further, we provide example parameter choices for the proposed scheme based on the LWE assumption and show that our scheme achieves competitive ciphertext sizes to that of SLAP [24] for equivalent plaintext spaces.

### 1.2 Related Work

The notion of PSA was introduced by Shi et al. in [21]. They proposed a construction based on the Decisional Diffie-Hellman (DDH) assumption. The decryption

procedure is inefficient due to its requirement for computing a discrete logarithm. Subsequent works [7,14,17] focused on constructing PSA with better efficiency and larger plaintext space. Leontiadis et al. introduced PSA with verifiability of the aggregated sum [18] followed by a construction by Emura [13]. These works are not post-quantum secure and can be broken easily by a quantum computer using Shor's algorithm [22].

A number of post-quantum PSA constructions have been proposed in previous works. Valovich proposed a PSA scheme from key homomorphic weak PRFs and gave an instantiation based on the LWE problem [26]. Their construction achieves a weaker variant of aggregator obliviousness (AO) called non-adaptive AO in the standard model. Further, the set of timestamps needs to be fixed at the setup and therefore their scheme only supports a bounded number of timestamps. Our scheme in the standard model follows a similar design policy as Valovich but we show how to get unbounded number of timestamps using a PRF. Becker et al. proposed a generic PSA scheme called LaPS [4] based on the LWE problem. Their construction can be instantiated using any additively homomorphic encryption scheme. However, their scheme uses two layers of encryption where the homomorphically encrypted input is encoded again using an Augmented-LWE (A-LWE) term. Further, their construction does not rely on timestamps directly and they only give a brief description on how to extend the scheme to work with timestamps. Takeshita et al. proposed two PSA schemes called SLAP [24] using two different fully homomorphic encryption schemes. Their schemes achieve aggregator obliviousness based on the RLWE problem in the random oracle model. The authors also implement their scheme and show their improvements over the LaPS protocol. In a subsequent work [23], Takeshita et al. proposed a variant of their SLAP protocol with better efficiency.

Other post-quantum secure works that do not use the RLWE problem include [15,27]. Ernst et al. proposed a PSA scheme using key-homomorphic PRFs [15] based on the Learning with Rounding (LWR) problem. Currently, this is one of the most efficient schemes that achieve smaller ciphertexts compared to previous works. Another efficient PSA scheme using labeled secret sharing schemes (LaSS) was proposed in [27]. However, it is not efficient for a large number of users due to multiple rounds of communication to generate shared keys among the users which leads to key sizes quadratic in the number of users.

All of these schemes rely on a trusted setup for key generation. There are brief discussions in [15,27] on how to modify their schemes to avoid a central authority. Recently, Brorsson et al. proposed a distributed setup PSA protocol called DIPSAUCE [10] that does not rely on a trusted party. Their protocol is a distributed setup variant of the protocol in [27]. In contrast to the other PSA schemes, no key is required for aggregating the sum of the inputs. However, their distributed key generation procedure relies on a Public Key Infrastructure (PKI) to provide the keys to each user which in turn is usually implemented as a central authority. Further DIPSAUCE relies on a randomness beacon and care should be taken not to introduce a trusted party to realize the beacon.

Another line of work focuses on constructing secure aggregation protocols for the aggregation of model updates in distributed machine learning [5,6,8,16,25]. These works are not directly comparable to ours as their work has a distinct focus, specifically designed to meet the requirements of distributed machine learning.

We give a comparison of the various PSA schemes described in this section with respect to different characteristics in Table 1.

**Table 1.** Comparison of different PSA schemes with respect to different characteristics

| Scheme | Decentralized Setup | Timestamps | Assumption | Post-quantum security |
|---|---|---|---|---|
| Shi et al. [21] | ✗ | unbounded | DDH | ✗ |
| Valovich [26] | ✗ | bounded | LWE | ✓ |
| LaPS [4] | ✗ | none | (R)LWE | ✓ |
| SLAP [24] | ✗ | unbounded | RLWE | ✓ |
| Ernst et al. [15] | ✗ | unbounded | LWR | ✓ |
| Waldner et al. [27] | ✗ | unbounded | security of LaSS | ✓ |
| DIPSAUCE [10] | ✓ | unbounded | security of LaSS | ✓ |
| Our Scheme | ✓ | unbounded | LWE | ✓ |

### 1.3 Organization

We organize the paper as follows. Section 2 contains some necessary background and definitions. In Sect. 3, we formally define the DPSA protocol and give a concrete construction based on the LWE problem in the ROM. In Sect. 4, we give a construction in the standard model based on LWE.

## 2 Preliminaries

**Notation**: $\lambda$ denotes the security parameter. For a set $S$, $a \leftarrow_{\$} S$ means that $a$ is sampled uniformly at random from $S$. For a probability distribution $\mathcal{X}$ over a set $S$, $x \leftarrow \mathcal{X}$ means that $x$ is sampled from $S$ according to the distribution $\mathcal{X}$. A distribution $\mathcal{X}$ over the set of integers is said to be $B$-bounded if it is supported on $[-B, B]$. For a number $x$, $\lceil x \rceil$, $\lfloor x \rfloor$ and $\lfloor x \rceil$ denotes the rounding $x$ up, down and to the closest integer respectively. We use 'log' to denote a logarithm to the base 2. For a prime $q$, $\mathbb{Z}_q$ denotes the set of integers in the interval $(-q/2, q/2] \cap \mathbb{Z}$. For some $a \in \mathbb{Z}$, we use $(a \bmod q)$ and $[a]_q$ interchangeably to denote the modular reduction of $a$ by $q$ into the interval $(-q/2, q/2] \cap \mathbb{Z}$. We use lowercase boldface letters (e.g., $\mathbf{a}$) to denote row vectors and uppercase boldface letters (e.g., $\mathbf{A}$) to

denote matrices. The notation $[n]$ denotes the set of integers $\{1, 2, \ldots, n\}$. An arbitrary negligible function is denoted by $\mathsf{negl}(\cdot)$ where the function $\mathsf{negl}(x) : \mathbb{N} \to \mathbb{R}$ is called negligible if for every $c \in \mathbb{N}$, there exists an integer $\eta_c$ such that $|\mathsf{negl}(x)| < \frac{1}{x^c}$ for all $x > \eta_c$.

### 2.1 Lattices

A $k$ dimensional lattice $\Lambda$ is a discrete additive subgroup of $\mathbb{R}^k$ given by the set of all integer linear combinations of $l \leq k$ linearly independent vectors in $\mathbb{R}^k$ where $l$ is called the rank of $\Lambda$. We are interested in $q$-ary integer lattices. A $q$-ary lattice can be thought of as a discrete additive subgroup of $\mathbb{Z}_q^k$. A vector $\mathbf{v}$ is in the lattice $\Lambda$ if $\mathbf{v} \bmod q \in \Lambda$. Given a matrix $\mathbf{B} \in \mathbb{Z}_q^{l \times k}$, the following are two $k$ dimensional $q$-ary lattices.

$$\Lambda_q(\mathbf{B}) = \left\{\mathbf{v} \in \mathbb{Z}_q^k \mid \mathbf{v} = \mathbf{w} \cdot \mathbf{B} \bmod q \text{ for some } \mathbf{w} \in \mathbb{Z}_q^l\right\}$$
$$\Lambda_q^{\perp}(\mathbf{B}) = \left\{\mathbf{v} \in \mathbb{Z}_q^k \mid \mathbf{v} \cdot \mathbf{B}^T = \mathbf{0} \bmod q\right\}$$

### 2.2 Learning with Errors

Learning with Errors is the problem of solving a system of noisy linear equations over $\mathbb{Z}_q$ [19]. It can be defined as follows.

**Definition 1 (Learning with errors).** *Let $\mathcal{X}$ be a probability distribution on $\mathbb{Z}$ and $\mathbf{s}$ be a secret vector chosen uniformly at random from $\mathbb{Z}_q^n$ for some $n, q \in \mathbb{N}$. Let $\mathcal{A}_{\mathbf{s},\mathcal{X}}$ be the distribution that generates a pair $(\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by choosing a vector $\mathbf{a} \leftarrow_{\$} \mathbb{Z}_q^n$ and an error $e \leftarrow \mathcal{X}$. Given polynomially many samples from $\mathcal{A}_{\mathbf{s},\mathcal{X}}$, the learning with errors problem is to output the vector $\mathbf{s} \in \mathbb{Z}_q^n$ with overwhelming probability.*

*The decisional LWE problem is to distinguish the distribution $\mathcal{A}_{\mathbf{s},\mathcal{X}}$ from the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$. We use $\mathrm{LWE}_{n,q,\mathcal{X}}$ to denote the LWE problem with parameters $n, q, \mathcal{X}$.*

The decisional LWE problem has been shown to be at least as hard as the LWE search problem [19,20]. There are known quantum and classical reductions of LWE to approximating short vector problems in lattices [9,20]. In these reductions, the noise distribution $\mathcal{X}$ is usually considered to be a discretized Gaussian distribution that is indistinguishable from a $B$-bounded distribution for some appropriate $B$.

The security of our protocol is based on a variant of the decisional LWE problem where along with the noise, the secret $\mathbf{s}$ is chosen from the distribution $\mathcal{X}$.

**Definition 2 (LWE problem with short secrets).** *Let $\mathcal{X}$ be a probability distribution on $\mathbb{Z}$ and $\mathbf{s}$ be a secret vector chosen from the distribution $\mathcal{X}$ over $\mathbb{Z}_q^n$ for some $n, q \in \mathbb{N}$. Let $\mathcal{A}_{\mathbf{s},\mathcal{X}}$ be the distribution defined in Definition 1. Then, the decision LWE problem with short secrets is to distinguish the distribution $\mathcal{A}_{\mathbf{s},\mathcal{X}}$ from the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$. We use $\text{ss-LWE}_{n,q,\mathcal{X}}$ to denote the LWE problem with short secrets.*

A reduction from the short secret variant exists to the decisional LWE problem as shown in [2].

**Lemma 1 ([2]).** *Let $n, q, \mathcal{X}$ be as described above. If there exists a distinguishing algorithm $\mathcal{A}$ for the decision LWE problem with short secrets, then there exists a distinguishing algorithm $\mathcal{B}$ for the decision LWE problem that runs in roughly the same time as $\mathcal{A}$, with $\mathcal{B}$ making $\mathcal{O}(n^2)$ calls to its oracle and satisfying $Adv_{\mathcal{B}}^{\mathsf{LWE}}(\lambda) = Adv_{\mathcal{A}}^{\mathsf{ss\text{-}LWE}}(\lambda)$.*

In this paper, we consider an extended form of the problem where the secret is a matrix. We consider the LWE distribution with $N \geq 1$ secrets $\mathbf{s}_1, \ldots, \mathbf{s}_N$ for some $N = poly(n)$. Then $\mathcal{A}_{\mathbf{S},\mathcal{X}}$ is defined as the distribution that generates a pair $(\mathbf{a}, \mathbf{b} := \mathbf{a} \cdot \mathbf{S}^\top + \mathbf{e})$ obtained by choosing $\mathbf{a} \leftarrow_\$ \mathbb{Z}_q^n$ and an error vector $\mathbf{e} \leftarrow_\$ \mathcal{X}^N$ where the $i$-th row of $\mathbf{S} \in \mathbb{Z}_q^{N \times n}$ is the secret $\mathbf{s}_i$. Using a standard hybrid argument, it can be shown that distinguishing $\mathbf{S}$ from uniformly random is as hard as the $\mathrm{LWE}_{n,q,\mathcal{X}}$ problem.

### 2.3 Pseudorandom Functions

A pseudorandom function (PRF) is an efficiently computable deterministic function that is computationally indistinguishable from a truly random function.

**Definition 3 (PRF).** *A pseudorandom function family $\mathcal{F} = \{F_K\}_{K \in \mathcal{K}_\lambda}$ with keyspace $\mathcal{K}_\lambda$ is a family of functions $F_K : \mathcal{X} \to \mathcal{Y}$ such that $F_K$ can be computed in $poly(\lambda)$ time and for any $x \in \mathcal{X}$, $F_K(x)$ cannot be distinguished from a random function (RF) in polynomial time. For all PPT adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in distinguishing a PRF from a RF is given by*

$$Adv_{\mathcal{A}}^{\mathsf{PRF}}(\lambda) = \left| Pr[\mathcal{A}^{F_K(\cdot)}(\lambda) = 1] - Pr[\mathcal{A}^{\mathsf{RF}(\cdot)}(\lambda) = 1] \right| \leq \mathsf{negl}(\lambda)$$

## 3 Decentralized Private Stream Aggregation

In this section, we formally define a decentralized PSA (DPSA) scheme and give a concrete construction based on the LWE problem. Consider a scenario with $\ell$ users for some $\ell \in \mathbb{N}$ and an untrusted aggregator. We consider the users to be semi honest, i.e., honest but curious. Each user generates private data $\mathbf{x}_{i,t}$ with respect to some time stamp $t$ and wishes to compute the sum $\sum_{i=1}^{\ell} \mathbf{x}_{i,t}$ securely and privately. In Private Stream Aggregation (PSA) the sum can be computed by the aggregator given only the encrypted values of the user's data while preserving the user's privacy. The users encrypt their data $\mathbf{x}_{i,t}$ using a user specific secret key $\mathsf{sk}_i$ before sending it to the aggregator. The aggregator then performs the aggregating function on the encrypted data and recovers the sum of the input data using an aggregator decryption key $\mathsf{dk}_0$. In a centralized PSA scheme, the encryption and decryption keys are generated by a trusted setup. Since the setup in DPSA is decentralized, the users need to generate the aggregator decryption

key themselves apart from their own encryption keys. Each user generates a share of the aggregator key and sends it to the aggregator in a secure way without revealing their individual keys. Upon receiving the partial keys from all the users, the aggregator can recover its decryption key for aggregation.

**Definition 4 (Decentralized Private Stream Aggregation).** *A decentralized private stream aggregation scheme over a message space $\mathcal{M}$ consists of the following PPT algorithms:*

$\mathsf{Setup}(1^\lambda, 1^\ell)$: *This is a procedure between the users. It takes the security parameter $\lambda$ and the number of users $\ell$ and generates the public parameters $\mathsf{pp}$ and their own secret keys $\mathsf{sk}_i$ for $i \in [\ell]$. The public parameters $\mathsf{pp}$ is an implicit input to the rest of the algorithms.*

$\mathsf{AggKeyGenShare}(i, \mathsf{sk}_i)$: *It takes user index $i$ and secret key $\mathsf{sk}_i$ and outputs the partial aggregator key $\mathsf{dk}_i$.*

$\mathsf{AggKeyGen}(\{\mathsf{dk}_i\}_{i\in[\ell]})$: *It takes the partial aggregator keys $\mathsf{dk}_i$ for $i \in [\ell]$ and computes aggregator decryption key $\mathsf{dk}_0 = \sum_{i=1}^{\ell} \mathsf{dk}_i$.*

$\mathsf{Enc}(i, \mathsf{sk}_i, \mathbf{x}_{i,t}, t)$: *It takes as input the user index $i$, the secret key $\mathsf{sk}_i$, timestamp $t$ and input data $\mathbf{x}_{i,t} \in \mathcal{M}$ and outputs a ciphertext $\mathsf{ct}_{i,t}$.*

$\mathsf{AggDec}(\mathsf{dk}_0, \{\mathsf{ct}_{i,t}\}_{i\in[\ell]}, t)$: *It outputs the aggregated sum $\mathbf{x}_t = \sum_{i=1}^{\ell} \mathbf{x}_{i,t}$ from the ciphertexts $\{\mathsf{ct}_{i,t}\}_{i\in[\ell]}$ using $\mathsf{dk}_0$ for the time period $t$.*

Here, the $\mathsf{Setup}$ algorithm is run between the users to generate the public parameters $\mathsf{pp}$ and their secret keys $\mathsf{sk}_i$. The users compute partial aggregator keys $\mathsf{dk}_i$ using $\mathsf{AggKeyGenShare}$ and sends $\mathsf{dk}_i$ to the aggregator. The aggregator computes its decryption key $\mathsf{dk}_0$ using $\mathsf{dk}_0 \leftarrow \mathsf{AggKeyGen}(\{\mathsf{dk}_i\}_{i\in[\ell]})$. Each user $i$ then encrypts their input $\mathbf{x}_{i,t}$ at timestamp $t$ such that $\mathsf{ct}_{i,t} \leftarrow \mathsf{Enc}(i, \mathsf{sk}_i, \mathbf{x}_{i,t}, t)$. The aggregator outputs $\mathbf{x}_t \leftarrow \mathsf{AggDec}(\mathsf{dk}_0, \{\mathsf{ct}_{i,t}\}_{i\in[\ell]}, t)$. The algorithms $\mathsf{Setup}$, $\mathsf{AggKeyGenShare}$ and $\mathsf{AggKeyGen}$ are run only once in the beginning of the protocol.

**Correctness**: The above DPSA scheme $\mathsf{DPSA}$=($\mathsf{Setup}, \mathsf{AggKeyGenShare}, \mathsf{AggKeyGen}, \mathsf{Enc}, \mathsf{AggDec}$) is said to be correct if for any $\lambda, \ell \in \mathbb{N}$, any message $\mathbf{x}_{i,t} \in \mathcal{M}$, it holds that

$$\Pr\left[\mathsf{AggDec}(\mathsf{dk}_0, \{\mathsf{ct}_{i,t}\}_{i\in[\ell]}, t) = \sum_{i=1}^{\ell} \mathbf{x}_{i,t} : \begin{array}{l} (\mathsf{pp}, \{\mathsf{sk}_i\}_{i\in[\ell]}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell) \\ \{\mathsf{dk}_i\}_{i\in[\ell]} \leftarrow \mathsf{AggKeyGenShare}(i, \mathsf{sk}_i) \\ \mathsf{dk}_0 \leftarrow \mathsf{AggKeyGen}(\{\mathsf{dk}_i\}_{i\in[\ell]}) \\ \mathsf{ct}_{i,t} \leftarrow \mathsf{Enc}(i, \mathsf{sk}_i, \mathbf{x}_{i,t}, t) \end{array}\right] = 1$$

**Security**: The security of a private stream aggregation scheme is captured by the notion of *aggregator obliviousness*. It requires that the aggregator does not learn anything more than the aggregated value of their input values at each time period. If some parties collude with the aggregator then it requires that the aggregator only learns the aggregated value of the honest users and nothing more. Further, each user encrypts their data only once every time period.

**Definition 5 (Aggregator Obliviousness for DPSA).** *The aggregator obliviousness security for a DPSA scheme can be defined in terms of the security experiment* $\mathsf{AO}_\beta(\lambda, \ell, \mathcal{A})$ *given in Fig. 1. No adversary* $\mathcal{A}$ *should be able to win this game with non-negligible advantage.*

$\mathsf{AO}_\beta(\lambda, \ell, \mathcal{A})$

1: $(\mathsf{pp}, \{\mathsf{sk}_i\}_{i\in[\ell]}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$
2: $\beta \leftarrow \mathcal{A}^{\mathsf{QCorr}(\cdot),\mathsf{QEnc}(\cdot,\cdot,\cdot,\cdot),\mathsf{QChallenge}(\cdot,\cdot,\cdot,\cdot)}(\mathsf{pp})$
3: **if** condition $(*)$ is satisfied **then**
4: output $\beta$
5: **else**
6: output 0

**Fig. 1.** Aggregator Obliviousness experiment for DPSA

*The challenger first runs the* Setup *algorithm and returns the public parameters* pp *to the adversary. The adversary makes queries to the following oracles:*

- ***Corruption oracle*** $\mathsf{QCorr}(i)$*: The adversary submits an integer* $i \in \{1, \ldots, \ell\}$ *and gets back the* $i$*-th user's secret key* $\mathsf{sk}_i$*. If the adversary submits* $i = 0$*, then it gets* $\mathsf{dk}_j \leftarrow \mathit{AggKeyGenShare}(j, \mathsf{sk}_j)$ *for all* $j \in [\ell]$*.*
- ***Encryption oracle*** $\mathsf{QEnc}(i, \mathbf{x}_{i,t}, t)$*: The adversary submits* $(i, \mathbf{x}_{i,t}, t)$ *and receives* $\mathsf{ct}_{i,t}, \leftarrow \mathit{Enc}(i, \mathsf{sk}_i, \mathbf{x}_{i,t}, t)$ *from the challenger.*
- ***Challenge oracle*** $\mathsf{QChallenge}(\mathcal{U}, \{\mathbf{x}^0_{i,t^*}\}_{i\in\mathcal{U}}, \{\mathbf{x}^1_{i,t^*}\}_{i\in\mathcal{U}}, t^*)$*: This query can be made only once by the adversary. The adversary selects a set of users* $\mathcal{U}$ *and time period* $t^*$ *and for each* $i \in \mathcal{U}$*, the adversary chooses two sets of inputs* $\mathbf{x}^0_{i,t^*}$*,* $\mathbf{x}^1_{i,t^*}$*. The challenger randomly samples* $b \leftarrow \{0, 1\}$ *and returns* $\mathsf{ct}_{i,t^*} \leftarrow \mathit{Enc}(\mathsf{sk}_i, \mathbf{x}^0_{i,t^*}, t^*)$ *for all* $i \in \mathcal{U}$ *if* $b = 0$ *and* $\mathsf{ct}_{i,t^*} \leftarrow \mathit{Enc}(\mathsf{sk}_i, \mathbf{x}^1_{i,t^*}, t^*)$ *for all* $i \in \mathcal{U}$ *if* $b = 1$*.*

*Finally, the adversary outputs a guess* $b'$ *for the value of* $b$ *and the experiment outputs* $\beta$ *depending on the following conditions.*

*Let* $\mathcal{CS}$ *be the set of corrupted users,* $\mathcal{HS}$ *be the set of honest users at the end of the game and let* $\mathcal{E}_{t^*}$ *be the set of users for which an encryption query has been made at time* $t^*$*. Let* $\mathcal{Q}_{t^*} := \mathcal{U} \cup \mathcal{E}_{t^*}$ *be the set of users for which* $\mathcal{A}$ *receives an encryption or a challenge ciphertext at timestamp* $t^*$*. The condition* $(*)$ *is satisfied if all of the following conditions hold:*

- $\mathcal{U} \cap \mathcal{CS} = \emptyset$*: The set of users specified during the* Challenge *phase must be uncorrupted at the end of the game.*
- *Adversary* $\mathcal{A}$ *has not queried* $\mathsf{QEnc}(i, \mathbf{x}_{i,t}, t^*)$ *for the same* $i$ *and* $t^*$*. Otherwise, this would violate the encrypt-once policy.*
- $\mathcal{U} \cap \mathcal{E}_{t^*} = \emptyset$*: The adversary cannot query challenge ciphertexts to the users in* $\mathcal{E}_{t^*}$*. In other words, the adversary cannot get a challenge ciphertext from users for which it has queried the encryption oracle at time* $t^*$*.*

– *If the adversary has compromised the aggregator and* $\mathcal{Q}_{t^*} \cup \mathcal{CS} = [\ell]$*, then the following condition must be satisfied.*

$$\sum_{i \in \mathcal{U}} \boldsymbol{x}^0_{i,t^*} = \sum_{i \in \mathcal{U}} \boldsymbol{x}^1_{i,t^*}$$

*We set* $\beta \leftarrow b'$ *if the above conditions are satisfied, otherwise we set* $\beta = 0$.

*A DPSA scheme is said to be aggregator oblivious if for any PPT adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}$ *such that*

$$\mathsf{Adv}^{\mathsf{AO}}_{\mathcal{A},\mathsf{DPSA}}(\lambda, \ell) = |Pr[\mathsf{AO}_0(\lambda, \ell, \mathcal{A}) = 1] - Pr[\mathsf{AO}_1(\lambda, \ell, \mathcal{A}) = 1]| \leq \mathsf{negl}(\lambda)$$

If an adversary can corrupt the parties only at the beginning of the protocol, then we say that the scheme is secure against *static corruptions*. On the other hand, if an adversary can corrupt the parties dynamically during the execution of the protocol, then we say that the scheme is secure against adaptive corruptions. For static security, the corruption queries are sent by the adversary before obtaining the public parameters.

### 3.1 Our Construction

Our concrete DPSA scheme over the plaintext space $\mathcal{M} := \mathbb{Z}_p^n$ can be described in terms of the following PPT algorithms.

$\mathsf{Setup}(1^\lambda, 1^\ell)$: This is a protocol between the users. Let $H$ be a hash function mapping from the domain of all timestamps onto $\mathbb{Z}_q^n$. Let $\mathcal{X}$ be a $B$-bounded distribution over $\mathbb{Z}$. Each user generates a matrix $\mathbf{S}_i \leftarrow_\$ \mathcal{X}^{n \times n}$ and interactively generates secret shares $\mathbf{V}_i \leftarrow \mathbb{Z}_q^{n \times n}$ of 0 such that $\sum_{i=1}^{\ell} \mathbf{V}_i = 0 \bmod q$. Output public parameters $\mathsf{pp} = (p, q, n, \ell, H, \mathcal{X})$ and user secret keys $\mathsf{sk}_i = (\mathbf{S}_i, \mathbf{V}_i)$ for $i \in [\ell]$. The public parameters $\mathsf{pp}$ is an implicit input to all the algorithms.

$\mathsf{AggKeyGenShare}(i, \mathsf{sk}_i)$: Given user index $i$ and secret key $\mathsf{sk}_i = (\mathbf{S}_i, \mathbf{V}_i)$, compute partial aggregator key $\mathsf{dk}_i = \mathbf{S}_i + \mathbf{V}_i \pmod q$.

$\mathsf{AggKeygen}(\{\mathsf{dk}_i\}_{i \in [\ell]})$: Given $\{\mathsf{dk}_i\}_{i \in [\ell]}$, compute aggregator decryption key

$$\mathsf{dk}_0 := \sum_{i=1}^{\ell} \mathsf{dk}_i = \sum_{i=1}^{\ell} (\mathbf{S}_i + \mathbf{V}_i) = \sum_{i=1}^{\ell} \mathbf{S}_i \pmod q = \mathbf{S}_0 \tag{1}$$

$\mathsf{Enc}(i, \mathsf{sk}_i, \mathbf{x}_{i,t}, t)$: Given input $\mathbf{x}_{i,t} \in \mathbb{Z}_p^n$ and timestamp $t$, sample $\mathbf{e}_{i,t} \leftarrow \mathcal{X}^n$. Set $\mathbf{y}_t := H(t) \in \mathbb{Z}_q^n$ and compute the ciphertext $\mathsf{ct}_{i,t}$ as

$$\mathsf{ct}_{i,t} = \mathbf{x}_{i,t} + \mathbf{y}_t \cdot \mathbf{S}_i^\top + p \cdot \mathbf{e}_{i,t} \pmod q \tag{2}$$

$\mathsf{AggDec}\big(\mathsf{dk}_0, \{\mathsf{ct}_{i,t}\}_{i \in [\ell]}, t\big)$: Compute $\mathbf{y}_t = H(t)$ and output the aggregated sum

$$\mathbf{x}_t = \left[ \left( \sum_{i=1}^{\ell} \mathsf{ct}_{i,t} - \mathbf{y}_t \cdot \mathbf{S}_0^\top \pmod q \right) \right]_p \tag{3}$$

**Correctness**: The correctness of the sum can be verified as follows:

$$\sum_{i=1}^{\ell} \mathsf{ct}_{i,t} - \mathbf{y}_t \cdot \mathbf{S}_0^{\top} \pmod q = \sum_{i=1}^{\ell} \mathbf{x}_{i,t} + p \cdot \sum_{i=1}^{\ell} \mathbf{e}_{i,t} \pmod q \tag{4}$$

The magnitude of the sum of the errors is bounded by $\ell \cdot p \cdot B$ where $B$ is the maximum bound on the error distribution $\mathcal{X}$. The magnitude of the sum of the inputs is bounded by $\ell \cdot \frac{p}{2}$. If $\frac{\ell \cdot p}{2}(1 + 2B) < \frac{q}{2}$, then $\sum_{i=1}^{\ell} \mathbf{x}_{i,t} + p \cdot \sum_{i=1}^{\ell} \mathbf{e}_{i,t} \pmod q = \sum_{i=1}^{\ell} \mathbf{x}_{i,t} + p \cdot \sum_{i=1}^{\ell} \mathbf{e}_{i,t}$ and reducing it modulo $p$ removes the error and recovers the sum $\sum_{i=1}^{\ell} \mathbf{x}_{i,t}$.

### 3.2 Aggregator Obliviousness

We show that the proposed construction achieves aggregator obliviousness with static corruptions in the encrypt-once security model under the hardness of the LWE problem.

**Theorem 1.** *For any PPT adversary $\mathcal{A}$ against the aggregator obliviousness game, there exists a PPT adversary $\mathcal{B}$ against the LWE problem such that*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{AO}}(\lambda, \ell) \leq (8\ell^3 + 4\ell^2) \cdot \mathsf{Adv}_{\mathcal{B}}^{\mathsf{LWE}}(\lambda)$$

*Proof.* We use similar ideas from [15] to prove this Theorem. WLOG, we assume that the adversary queries the $\mathsf{QChallenge}$ oracle only at one timestamp $t^*$ that has not been queried to the $\mathsf{QEnc}$ oracle.

We proceed via a series of Games $\mathsf{G}_i$ for $i \in \{0, 1, 2, 3\}$ described in Fig. 4 of Appendix B. A summary of the transitions is provided in Table 2. We denote the advantage of $\mathcal{A}$ in game $\mathsf{G}_i$ using $\mathsf{Adv}_{\mathcal{A}}(\mathsf{G}_i)$. Similar to [15], we consider two cases. I) *When the adversary corrupts the aggregator*: The adversary can decrypt the sum in this case and therefore, we need to make sure that the sum remains unchanged throughout the games. II) *When the adversary does not corrupt the aggregator*: In this case we can directly go from game $\mathsf{G}_0$ to $\mathsf{G}_3$ using a hybrid argument over all the users.

**Game** $\mathsf{G}_0$: This is the $\mathsf{AO}_0$ game where the challenge query is answered with the encryption of $\mathbf{x}_{i,t}^0$.

**Game** $\mathsf{G}_1$: In Game $\mathsf{G}_1$, we change the way the vectors $\mathbf{c}_{i,t}$ in $\mathsf{QChallenge}$ are generated. The challenge query is still answered with encryptions of $\mathbf{x}_{i,t}^0$ but we add a share of a perfect $\mu$-out-of-$\mu$ secret sharing of zero denoted by $\mathbf{r}_i \leftarrow SS(0)$ to $\mathbf{c}_{i,t}$s where $\mu$ is the number of users in the challenge query. We need to make this change in such a way that the aggregate sum on decryption remains the same. The transition from $\mathsf{G}_0$ to $\mathsf{G}_1$ can be proved via a hybrid argument over the $\ell$ users relying on the LWE assumption.

**Lemma 2 (Transition from $\mathsf{G}_0$ to $\mathsf{G}_1$).** *For all PPT adversary $\mathcal{A}$, that corrupts the aggregator, there exists a PPT adversary $\mathcal{B}$ such that*

$$|\mathsf{Adv}_{\mathcal{A}}(\mathsf{G}_0) - \mathsf{Adv}_{\mathcal{A}}(\mathsf{G}_1)| \leq 2\ell h(h-1) \cdot \mathsf{Adv}_{\mathcal{B}}^{\mathsf{LWE}}(\lambda)$$

**Table 2.** A summary of the games used in the proof of Theorem 1. Change in each game is highlighted with a square box

| Game | $\mathsf{ct}_{i,t}$ | Justification |
|---|---|---|
| $\mathsf{G}_0$ | $\mathbf{c}_{i,t} \leftarrow \mathbf{y}_t \cdot \mathbf{S}_i + p \cdot \mathbf{e}_{i,t}$<br>$\mathsf{ct}_{i,t} \leftarrow \mathbf{x}^0_{i,t} + \mathbf{c}_{i,t}$ | $\mathsf{AO}_0$ game |
| $\mathsf{G}_1$ | $\mathbf{c}'_{i,t} \leftarrow \mathbf{y}_t \cdot \mathbf{S}_i + p \cdot \mathbf{e}_{i,t}$<br>$\boxed{\mathbf{c}_{i,t} \leftarrow \mathbf{c}'_{i,t} + \mathbf{r}_i}, \quad \mathbf{r}_i \leftarrow SS(0)$<br>$\mathsf{ct}_{i,t} \leftarrow \mathbf{x}^0_{i,t} + \mathbf{c}_{i,t}$ | LWE assumption |
| $\mathsf{G}_2$ | $\mathbf{c}'_{i,t} \leftarrow \mathbf{y}_t \cdot \mathbf{S}_i + p \cdot \mathbf{e}_{i,t}$<br>$\mathbf{c}_{i,t} \leftarrow \mathbf{c}'_{i,t} + \mathbf{r}_i, \quad \mathbf{r}_i \leftarrow SS(0)$<br>$\boxed{\mathsf{ct}_{i,t} \leftarrow \mathbf{x}^1_{i,t} + \mathbf{c}_{i,t}}$ | information-theoretic |
| $\mathsf{G}_3$ | $\boxed{\mathbf{c}_{i,t} \leftarrow \mathbf{y}_t \cdot \mathbf{S}_i + p \cdot \mathbf{e}_{i,t}}$<br>$\mathsf{ct}_{i,t} \leftarrow \mathbf{x}^1_{i,t} + \mathbf{c}_{i,t}$ | LWE assumption |

*Proof.* To prove this transition, we use a sequence of hybrid games $\mathsf{G}_{0.l}$ for $l \in [\ell]$ defined in Fig. 5 of Appendix B. Note that, $\mathsf{G}_0 := \mathsf{G}_{0.1}$ and $\mathsf{G}_1 := \mathsf{G}_{0.\ell}$. The goal in each hybrid game is to add a perfect secret share of 0 to the LWE mask $\mathbf{c}_{i,t} := \mathbf{y}_t \cdot \mathbf{S}_i^\top + p \cdot \mathbf{e}_{i,t}$ of one more user. Let $\mu := |\mathcal{U}|$ where $\mathcal{U} := \{i_1, \ldots, i_\mu\}$ is the set of users specified by $\mathcal{A}$ during $\mathsf{QChallenge}$. Let $K := min(\mu, l)$. If $K \geq 2$ in hybrid step $l$, then a share of a perfect $K$-out-of-$K$ secret sharing of 0 is added to the LWE masks of the first $K$ users in $\mathcal{U}$. This can be done using two users at a time and the condition $K \geq 2$ is needed to go from one hybrid to another. To prove the indistinguishability of $\mathsf{G}_1$ from $\mathsf{G}_0$, it suffices to show that the adjacent games $\mathsf{G}_{0.l-1}$ and $\mathsf{G}_{0.l}$ are computationally indistinguishable. Precisely, we have

$$|\mathsf{Adv}_{\mathcal{A}}(\mathsf{G}_0) - \mathsf{Adv}_{\mathcal{A}}(\mathsf{G}_1)| = \sum_{l=1}^{\ell} |\mathsf{Adv}_{\mathcal{A}}(\mathsf{G}_{0.l-1}) - \mathsf{Adv}_{\mathcal{A}}(\mathsf{G}_{0.l})|$$

If there is an adversary $\mathcal{A}$ that can distinguish $\mathsf{G}_{0.l-1}$ from $\mathsf{G}_{0.l}$, then there exists an adversary $\mathcal{B}$ against the $\mathrm{LWE}_{n,q,\mathcal{X}}$ assumption. We consider the case $K \geq 2$. In $\mathsf{G}_{0.l-1}$, we have secret shares added to $\mathbf{c}_{i,t} = \mathbf{y}_t \cdot \mathbf{S}_i^\top + p \cdot \mathbf{e}_{i,t}$ of the first $K-1$ users in $\mathcal{U}$. To add a share of a perfect $K$-out-of-$K$ secret sharing of 0 to the $K$-th user in $\mathcal{U}$, $\mathcal{B}$ first guesses the first and the $K$-th users of $\mathcal{U}$ such that $i_1^* \leftarrow_\$ [\mathcal{HS}]$, $i_K^* \leftarrow_\$ [\mathcal{HS}] \setminus \{i_1^*\}$ where $\mathcal{HS} = [\ell] \setminus \mathcal{CS}$ is the set of honest users. $\mathcal{B}$ then samples $\mathbf{S}_i \leftarrow_\$ \mathcal{X}^{n \times n}$ and $\mathbf{V}_i \leftarrow_\$ \mathbb{Z}_q^{n \times n}$ for $i \in [\ell] \setminus \{i_1^*, i_K^*\}$. It can therefore set $\mathsf{sk}_i := (\mathbf{S}_i, \mathbf{V}_i)$ for $i \in \mathcal{CS}$ and send them to $\mathcal{A}$. It also samples aggregator key $\mathbf{S}_0 \leftarrow \mathcal{X}^{n \times n}$ uniformly at random. If the guess is incorrect, the simulation aborts the game and outputs 0. If the guess is correct then it replaces $\mathbf{c}_{i_1^*,t}$ with a random vector $\mathbf{b}_t \leftarrow_\$ \mathbb{Z}_q^n$ using the LWE assumption on $\mathbf{S}_{i_1^*}$. To make sure that the sum $\mathbf{S}_0 = \sum_{i=1}^{\ell} \mathbf{S}_i$ is satisfied, we need to modify $\mathbf{c}_{i_K^*,t}$ as $\mathbf{c}_{i_K^*,t} := H(t) \cdot \mathbf{S}_0 - H(t) \sum_{j \in [\ell] \setminus \{i_1^*, i_K^*\}} \mathbf{S}_j - \mathbf{b}_t$. Then $\mathbf{c}_{i_1^*,t}$ and $\mathbf{c}_{i_1^*,t} + \mathbf{u}_K$ are

indistinguishable where $\mathbf{u}_K \leftarrow_{\$} \mathbb{Z}_q^n$. Then, replace $\mathbf{c}_{i_1^*,t}$ back with $\mathbf{y}_t \cdot \mathbf{S}_{i_1^*}^\top + p \cdot \mathbf{e}_{i_1^*,t}$ using the LWE assumption on $\mathbf{S}_{i_1^*}$.

The guessing of the users $i_1^*$ and $i_K^*$ incurs a security loss of $h(h-1)$ where $h$ is the number of users in $\mathcal{HS}$. Therefore for all $l \in \{2, \ldots, \ell]$ there exists a PPT adversary $\mathcal{B}$ such that

$$|\mathsf{Adv}_{\mathcal{A}}(\mathsf{G}_{0.l-1}) - \mathsf{Adv}_{\mathcal{A}}(\mathsf{G}_{0.l})| \leq h(h-1) \cdot \mathsf{Adv}_{\mathcal{B}}^{\mathsf{LWE}}$$

Summing up for all the hybrid games, it leads to a security loss of $\ell h(h-1)$. Since the reduction is applied twice, total loss is $2\ell h(h-1)$. Therefore, we obtain a PPT adversary $\mathcal{B}$ such that

$$|\mathsf{Adv}_{\mathcal{A}}(\mathsf{G}_0) - \mathsf{Adv}_{\mathcal{A}}(\mathsf{G}_1)| \leq 2\ell h(h-1) \cdot \mathsf{Adv}_{\mathcal{B}}^{\mathsf{LWE}}$$

□

Now, we are in game $\mathsf{G}_1$ and $\mathsf{QChallenge}(\mathcal{U}, \{\mathbf{x}_{i,t^*}^0\}_{i\in\mathcal{U}}, \{\mathbf{x}_{i,t^*}^1\}_{i\in\mathcal{U}}, t^*)$ in $\mathsf{G}_1$ is answered with $\mathbf{x}_{i,t^*}^0 + \mathbf{c}_{i,t^*} + \sum_{j\in\mathcal{U}\setminus\{i_1\}} \mathbf{u}_j$ for $i = i_1$ and $\mathbf{x}_{i,t^*}^0 + \mathbf{c}_{i,t^*} - \mathbf{u}_i$ for $i \in \mathcal{U} \setminus \{i_1\}$. This is clear that these shares form a perfect $\mu$ out of $\mu$ secret sharing of 0. Further, the corruption queries in $\mathsf{G}_1$ are answered as follows. On input $i \in [\mathcal{CS}]$, $\mathcal{B}$ returns the key $\mathsf{sk}_i$ to $\mathcal{A}$. If the adversary corrupts the aggregator, then $\mathsf{QCorr}$ queries are answered with partial decryption keys for the honest users because the keys for the corrupted users can be generated by the adversary itself. To answer $\mathsf{QCorr}(0)$, $\mathcal{B}$ first generates secret shares of 0 for all the honest users, $\mathbf{R}_i \leftarrow SS(0)$ and computes

$$\begin{aligned}
\mathsf{dk}_i &= \mathbf{S}_i + \mathbf{V}_i + \mathbf{R}_i \quad \text{for } i \in \mathcal{HS} \setminus \{i_1^*, i_K^*\} \\
\mathsf{dk}_{i_1^*} &= \mathbf{S}_0 - \sum_{j\in[\mathcal{CS}]} (\mathbf{S}_j + \mathbf{V}_j) + \mathbf{R}_{i_1 *} \\
\mathsf{dk}_{i_K^*} &= - \sum_{j\in\mathcal{HS}\setminus\{i_1^*, i_K^*\}} (\mathbf{S}_j + \mathbf{V}_j) + \mathbf{R}_{i_K^*}
\end{aligned}$$

**Game** $\mathsf{G}_2$: In this game, all the challenge queries are answered with encryptions of $\mathbf{x}_{i,t}^1$ instead of $\mathbf{x}_{i,t}^0$. This is possible because the secret shares hide all the information on the individual ciphertexts.

**Lemma 3 (Transition from $\mathsf{G}_1$ to $\mathsf{G}_2$).** *For all PPT adversary $\mathcal{A}$, that corrupts the aggregator, there exists a PPT adversary $\mathcal{B}$ such that*

$$|\mathsf{Adv}_{\mathcal{A}}(\mathsf{G}_1) - \mathsf{Adv}_{\mathcal{A}}(\mathsf{G}_2)| \leq 2 \cdot \mathsf{Adv}_{\mathcal{B}}^{\mathsf{LWE}}(\lambda)$$

*Proof.* Let $\mathcal{Q}_{t^*}$ be the set of users for which $\mathcal{A}$ has a ciphertext at timestamp $t^*$ and let $\mathcal{HS}$ be the set of honest users. We consider the following two cases here.

**Case 1** ($\mathcal{Q}_{t^*} = \mathcal{HS}$): In this case, the adversary receives a ciphertext for all the honest users at timestamp $t^*$ either from the encryption oracle or from the challenge oracle. Then $\mathcal{Q}_{t^*} \cup \mathcal{CS} = [\ell]$ and the condition $\sum_{i\in\mathcal{U}} \mathbf{x}_{i,t^*}^0 = \sum_{i\in\mathcal{U}} \mathbf{x}_{i,t^*}^1$

must be satisfied. Let $\mathbf{r}_i$ be the pads added to the ciphertexts of the users in $\mathcal{U}$ at the end of game $\mathsf{G}_1$, where

$$\mathbf{r}_i = \begin{cases} \sum\limits_{i \in \mathcal{U} \setminus \{i_1\}} \mathbf{u}_i & \text{if } \; i = i_1 \\ -\mathbf{u}_i & \text{if } \; i \in \mathcal{U} \setminus \{i_1\} \end{cases} \tag{5}$$

These $\mathbf{r}_i$'s are perfect secret shares of 0. Therefore $\{\mathbf{x}^0_{i,t} + \mathbf{c}_{i,t} + \mathbf{r}_i\}_{i \in \mathcal{U}}$ and $\{\mathbf{x}^1_{i,t} + \mathbf{c}_{i,t} + \mathbf{r}_i\}_{i \in \mathcal{U}}$ are perfect secret shares of $\sum_{i \in \mathcal{U}} (\mathbf{x}^0_{i,t} + \mathbf{c}_{i,t})$ and $\sum_{i \in \mathcal{U}} (\mathbf{x}^1_{i,t} + \mathbf{c}_{i,t})$ respectively. Since, $\sum_{i \in \mathcal{U}} \mathbf{x}^0_{i,t^*} = \sum_{i \in \mathcal{U}} \mathbf{x}^1_{i,t^*}$, $\{\mathbf{x}^0_{i,t} + \mathbf{c}_{i,t} + \mathbf{r}_i\}_{i \in \mathcal{U}}$ and $\{\mathbf{x}^1_{i,t} + \mathbf{c}_{i,t} + \mathbf{r}_i\}_{i \in \mathcal{U}}$ are perfect secret shares of the same secret and are therefore perfectly indistinguishable from each other.

**Case 2** ($\mathcal{Q}_{t^*} \neq \mathcal{HS}$): In this case, there exists an honest user from which the adversary does not get a ciphertext at timestamp $t^*$. Therefore the condition $\sum_{i \in \mathcal{U}} \mathbf{x}^0_{i,t^*} = \sum_{i \in \mathcal{U}} \mathbf{x}^1_{i,t^*}$ does not hold in this case. Since $\mathcal{HS}$ is known in advance, it is possible to identify an user in $\mathcal{HS} \setminus \mathcal{Q}_{t^*}$ that is not in $\mathcal{U}$. $\mathcal{B}$ then chooses two such users $i_h \in \mathcal{HS} \setminus \mathcal{Q}_{t^*}$ and $i_u \in \mathcal{U}$ and simulates the ciphertexts as follows. For $i = i_u$, $\mathcal{B}$ sets $\mathbf{c}_{i,t} = \mathbf{b}_t$ where $\mathbf{b}_t$ is a random vector in $\mathbb{Z}_q^n$. For $i = i_h$, $\mathcal{B}$ sets $\mathbf{c}_{i,t} = H(t) \cdot \mathbf{S}_0 - \sum_{i \in [\ell] \setminus i_h} H(t) \cdot \mathbf{S}_i + \mathbf{e}_{i,t}$. Next, we change the challenge queries from encryption of $\mathbf{x}^0_{i,t}$ to encryptions of $\mathbf{x}^1_{i,t}$. For $b \in \{0,1\}$, we have $\sum_{i \in \mathcal{U}} (\mathbf{x}^b_{i,t} + \mathbf{c}_{i,t}) = \sum_{i \in \mathcal{U} \setminus i_u} (\mathbf{x}^b_{i,t} + \mathbf{c}_{i,t}) + \mathbf{x}^b_{i_u,t} + \mathbf{c}_{i_u,t}$. Since $\mathbf{c}_{i_u,t}$ is a random vector in $\mathbb{Z}_q^n$, $\{\mathbf{x}^0_{i,t} + \mathbf{c}_{i,t} + \mathbf{r}_i\}_{i \in \mathcal{U}}$ and $\{\mathbf{x}^1_{i,t} + \mathbf{c}_{i,t} + \mathbf{r}_i\}_{i \in \mathcal{U}}$ are secret shares of a random value. Therefore, they are indistinguishable from each other. Finally, we change the random vector with an LWE mask again.

**Game** $\mathsf{G}_3$: In this game we remove the secret shares from the challenge ciphertexts. Therefore, this game is identical to $\mathsf{AO}_1$ where the challenge queries are answered with encryptions of $\mathbf{x}^1_{i,t}$.

**Lemma 4 (Transition from $\mathbf{G}_2$ to $\mathbf{G}_3$).** *For all PPT adversary $\mathcal{A}$, that corrupts the aggregator, there exists a PPT adversary $\mathcal{B}$ such that*

$$|\mathsf{Adv}_{\mathcal{A}}(\mathsf{G}_2) - \mathsf{Adv}_{\mathcal{A}}(\mathsf{G}_3)| \leq 2\ell h(h-1) \cdot \mathsf{Adv}^{\mathsf{LWE}}_{\mathcal{B}}(\lambda)$$

*Proof.* This is symmetric to the transition from $\mathsf{G}_0$ to $\mathsf{G}_1$ applying the changes backwards.

For the case when the adversary does not corrupt the aggregator, we can directly go from $\mathsf{G}_0$ to $\mathsf{G}_3$.

**Lemma 5 (Transition from $\mathbf{G}_0$ to $\mathbf{G}_3$).** *For all PPT adversaries $\mathcal{A}$, that do not corrupt the aggregator, there exists a PPT adversary $\mathcal{B}$ such that*

$$|\mathsf{Adv}_{\mathcal{A}}(\mathsf{G}_0) - \mathsf{Adv}_{\mathcal{A}}(\mathsf{G}_3)| \leq 2\ell h \cdot \mathsf{Adv}^{\mathsf{LWE}}_{\mathcal{B}}(\lambda)$$

*Proof.* In this case, the adversary does not corrupt the aggregator and we can directly go from $\mathsf{G}_0$ to $\mathsf{G}_3$ using a hybrid argument over all the users. Let $\mathcal{U} := \{i_1, \ldots, i_\mu\}$ be the set of users specified in the challenge phase. The hybrid game $\mathsf{H}_l$ is given by

$$\mathsf{H}_l: \quad \mathbf{c}_{i,t^*} = \begin{cases} \mathsf{Enc}(i, \mathbf{x}^0_{i,t^*}, t^*) & \text{if } i = i_\tau \text{ for } \tau > l \\ \mathsf{Enc}(i, \mathbf{x}^1_{i,t^*}, t^*) & \text{if } i = i_\tau \text{ for } \tau \leq l \end{cases}$$

In other words, in $\mathsf{H}_l$, the challenge query is answered with encryptions of $\mathbf{x}^1_{i,t^*}$ for $i \in \{i_1, \ldots, i_l\}$ and with encryptions of $\mathbf{x}^0_{i,t^*}$ for the rest of the users. Note that $\mathsf{G}_0 = \mathsf{H}_0$ and $\mathsf{G}_3 = \mathsf{H}_\ell$. It suffices to show that the adjacent games $\mathsf{H}_{l-1}$ and $\mathsf{H}_l$ are computationally indistinguishable. Let $\mathcal{A}$ be an adversary that can distinguish $\mathsf{H}_{l-1}$ and $\mathsf{H}_l$. Then there exists an adversary $\mathcal{B}$ against the LWE problem. In $\mathsf{H}_{l-1}$, the challenge query for users $i_\tau$ with $\tau \leq l-1$ is answered with encryptions of $\mathbf{x}^1_{i_\tau,t^*}$ and for users $i_\tau$ with $\tau > l-1$, it is answered with encryptions of $\mathbf{x}^0_{i_\tau,t^*}$. The simulation $\mathcal{B}$ first guesses the user $i_l \leftarrow_\$ [\mathcal{HS}]$ and replaces $\mathbf{c}_{i,t^*} = H(t^*) \cdot \mathbf{S}_i^\top + p \cdot \mathbf{e}_{i,t^*}$ for $i = i_l$ with a random vector $\mathbf{b}_{t^*}$ using the LWE assumption on $\mathbf{S}_i$. Then $\mathbf{x}^0_{i,t^*} + \mathbf{c}_{i,t^*}$ is computationally indistinguishable from $\mathbf{x}^1_{i,t^*} + \mathbf{c}_{i,t^*}$ for $i = i_l$. Then, change $\mathbf{c}_{i,t^*}$ back to $\mathbf{c}_{i,t^*} = H(t^*) \cdot \mathbf{S}_i^\top + p \cdot \mathbf{e}_{i,t^*}$ for $i = i_l$.

The guessing of the user $i_l$ incurs a loss of $h$ where $h$ is the number of uncompromised users and this leads to $\ell h$ for $\ell$ hybrid games. Total loss in this case is $2\ell h$. Therefore, there is a PPT adversary $\mathcal{B}$ such that

$$|\mathsf{Adv}_{\mathcal{A}}(\mathsf{G}_0) - \mathsf{Adv}_{\mathcal{A}}(\mathsf{G}_3)| \leq 2\ell h \cdot \mathsf{Adv}^{\mathsf{LWE}}_{\mathcal{B}}(\lambda)$$

□

□

### 3.3 Parameters

In this section, we describe how to choose parameters for the proposed scheme for correctness and security. The LWE problem is parameterised by $n, q, \mathcal{X}$ where $\mathcal{X}$ is a discrete Gaussian distribution with mean 0 and standard deviation $\sigma$. The choice of $n, q, \sigma$ determines the security level of the scheme. For correctness, we need $\frac{\ell \cdot p}{2}(1 + 2B) < \frac{q}{2}$.

We use the LWE estimator [1] and the condition for correctness to determine parameters for a security level of 128 bits. Given $n$, modulus $q$ is determined for an error distribution with standard deviation $\sigma = 3.2$. We give example parameters for 128 bit security level in Table 3 when the secret is sampled from the error distribution.

Further, we compare the size of the ciphertexts between our DPSA scheme and the noise scaled version of SLAP as shown in Table 4. For a smaller number of users, the ciphertext size of the proposed DPSA scheme is either the same as or smaller than that of the SLAP scheme. However, for a larger number of users, the SLAP scheme has a slightly better ciphertext size compared to the proposed DPSA scheme.

**Table 3.** Example parameters for the DPSA scheme with LWE dimension $n$, modulus $q$ and noise distribution with standard deviation $\sigma = 3.2$ for 128-bit security level for varying number of users $\ell$ and plaintext modulus $p$

| No. of users | $\log p$ | $n$ | $\log q$ | Ciphertext bytes |
|---|---|---|---|---|
| 100 | 16 | 1200 | 29 | 4350 |
| 1000 | 16 | 1400 | 31 | 5425 |
| 10000 | 32 | 2510 | 51 | 16001 |
| $10^{13}$ | 32 | 4892 | 80 | 48920 |
| $10^{15}$ | 128 | 13800 | 183 | 315675 |
| $10^{21}$ | 128 | 17300 | 203 | 438987 |

**Table 4.** Comparison of ciphertext size between SLAP and our DPSA scheme

| No. of users | $\log p$ | $\log q$ | | Ciphertext bytes | |
|---|---|---|---|---|---|
| | | $\text{SLAP}_{NS}$ | DPSA | $\text{SLAP}_{NS}$ | DPSA |
| 1000 | 16 | 28 | 31 | 16384 | 5425 |
| 10000 | 32 | 48 | 51 | 16384 | 16001 |
| $10^{15}$ | 128 | 184 | 183 | 196608 | 315675 |
| $10^{21}$ | 128 | 204 | 203 | 262144 | 438987 |

### 3.4 Decentralized Setup

In the proposed DPSA construction, the setup is an interactive protocol between the users who generate their own keys and share it with the aggregator in a secure way. The aggregator then recovers the aggregate key for decryption which is the sum of the user keys. The users can generate their keys by sampling $\mathbf{S}_i$ uniformly at random from $\mathcal{X}^{n\times n}(\mathbb{Z}_q)$ and setting $\mathsf{sk}_i = \mathbf{S}_i$ for $i \in [\ell]$. To share the key with the aggregator, each user adds a random pad to their key which when added sums to zero. These random pads can be generated using a secret sharing protocol among the users. Each user $U_i$ generates secret shares $\{\mathbf{V}_{i,1}, \ldots, \mathbf{V}_{i,\ell}\}$ of 0 and shares $\mathbf{V}_{i,j}$ with user $U_j$ for $j \in [\ell] \setminus \{i\}$. User $U_i$ then generates its pad as $\mathbf{V}_i = \sum_{j=1}^{\ell} \mathbf{V}_{j,i}$ for $i \in [\ell]$ which is added to its secret key and the partial key $\mathbf{S}_i + \mathbf{V}_i$ is sent to the aggregator. When these partial keys are added together, the $\mathbf{V}_i$s sum to zero and the aggregator recovers $\mathbf{S}_0 = \sum_{i=1}^{\ell} \mathbf{S}_i$.

The communication cost per client during setup is sending one share to every other user and sending the partial key to the aggregator. The computational cost involves generating its share $\mathbf{V}_i$ and computing the partial aggregator key $\mathsf{dk}_i$. The setup is executed only once in the beginning of the protocol and does not affect the overall performance of the scheme.

### 3.5 Client Failures

If a client fails to submit its input message, then the aggregator cannot evaluate the sum because the equation $\mathbf{S}_0 = \sum_{i=1}^{\ell} \mathbf{S}_i$ does not satisfy (because of the missing ciphertext) and the decryption outputs a random value. Chan et al. [11] proposed a generic solution to deal with this problem and it is applicable to all PSA schemes. They use differential privacy and allow the aggregator to learn partial sums of the user's inputs such that the total sum can always be computed for the non-failing clients.

Their idea is to use a binary tree where the leaf nodes represent the clients and the intermediate nodes represent the partial sums of the clients beneath that node. Technically, the aggregator and the clients run an instance of the PSA protocol for each intermediate node. Therefore, each client generates $\log \ell$ ciphertexts using $\log \ell$ secret keys corresponding to the number of nodes from the client to the root of the binary tree. The aggregator is given an aggregator key for each intermediate node. The aggregator will always be able to compute the sum for the non-failing clients, albeit with an increase in noise in the overall sum. For example, consider the binary tree in Fig. 2 [11] for $\ell = 8$. The notation $[i, j]$ denotes the sum of the inputs of clients $\{i, \ldots, j\}$. If client 4 fails, the aggregator fails to obtain the sums $[4, 4]$, $[3, 4]$ and $[1, 4]$. The aggregator then uses the blocks corresponding to the black nodes in the tree to compute the sum of the remaining clients.

### 3.6 Optimizing Peer-to-Peer Communication

As a byproduct of the fault tolerance technique, we can also use the binary tree to reduce peer-to-peer communication among the clients during the setup phase. Instead of generating secret shares for all the $\ell - 1$ clients, each client can now generate shares only for those clients with whom they share an intermediate node. This will reduce the communication cost per client during the setup phase.

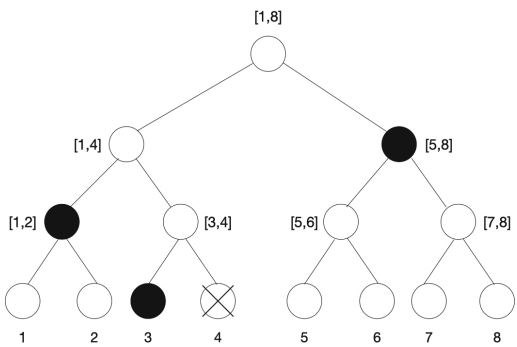


**Fig. 2.** When client 4 fails, the aggregator uses the partial sums corresponding to the black nodes (Color figure online)

### 3.7 Dynamic Join and Leave

Dynamic Join: Chan et al. [11] proposed the idea to create a tree with more leaf nodes than the number of clients to accommodate future client joining. In a centralized scheme, the trusted setup generates secret keys for every leaf node. The additional clients that have not joined the protocol yet are considered as failed until they join. Once a new client joins it receives a secret key from the setup. However, the trusted party needs to be present when a new client joins. In our DPSA scheme, we can use this technique as follows. When a new client $U_{\ell+1}$ joins the protocol before the computation of a new sum, the client first generates a uniformly random $\mathbf{S}_{\ell+1} \in \mathcal{X}^{n\times n}$ and sets $\mathbf{S}_{\ell+1}$ as its secret key. The client can broadcast its joining to the other clients through a bulletin board. Then each client that shares an intermediate node with the new client, chooses a new secret key and generates secret shares of zero and send these shares to the other clients that they share a node with. Using these shares, the clients then generate new aggregator keys and shares them with the aggregator. This is done for all the $\log \ell$ nodes.

*Dynamic Leave:* If some clients leave the protocol before the evaluation of a new sum, we can consider them as permanently failed. For the remaining clients, one possible solution is to run the Setup again. This will update their pads $\mathbf{V}_i$, which now consist of shares from the remaining users. Similarly, the aggregator receives a new key consisting of partial keys from the remaining users. Since the setup is decentralized, the users do not need to depend on a trusted entity to generate the updated pads or the updated aggregator key which makes it more practical than having a centralized setup.

## 4 DPSA in the Standard Model

In this section, we give a possible construction of a DPSA scheme in the standard model. We use similar ideas from [26] that uses a weak PRF to construct a PSA scheme based on the LWE problem. However, in [26], the number of timestamps is bounded as it needs to be fixed in the setup phase. We show how to get unbounded timestamps using a PRF. Let $\mathcal{F}_1 := \{F_{\mathbf{S}} \mid F_{\mathbf{S}} : \mathbb{Z}_q^n \to \mathbb{Z}_q^n, \mathbf{S} \in \mathbb{Z}_q^{n\times n}\}$ such that $F_{\mathbf{S}}(\mathbf{t}) = \mathbf{t} \cdot \mathbf{S}^\top + \mathbf{e}$. Here $\mathcal{F}_1$ is a randomized weak pseudorandom function family as described in [2,3]. Let $\mathcal{F}_2 = \{F_K \mid F_K : \mathbb{Z} \to \mathbb{Z}_q^n, K \in \mathcal{K}_\lambda\}$ be a PRF family such that $F_K(i) = \mathbf{t}_i \in \mathbb{Z}_q^n$. Then, a DPSA scheme in the standard model can be described in terms of the following algorithms.

$\mathsf{Setup}(1^\lambda, 1^\ell)$: This is a protocol between the users. Each user generates a matrix $\mathbf{S}_i \leftarrow_\$ \mathbb{Z}_q^{n\times n}$ and interactively generates $\mathbf{V}_i \leftarrow \mathbb{Z}_q^{n\times n}$ such that $\sum_{i=1}^{\ell} \mathbf{V}_i = 0 \bmod q$. Choose PRF key $K \leftarrow \mathcal{K}_\lambda$ and output public parameters $\mathsf{pp} = (p, q, n, \ell, K, \mathcal{X})$ and each user's secret key $\mathsf{sk}_i = (\mathbf{S}_i, \mathbf{V}_i)$. Since the PRF key is a public information, one of the clients can choose this key and broadcast it to the other clients.

$\mathsf{AggKeyGenShare}(i, \mathsf{sk}_i)$: Given user index $i$ and secret key $\mathsf{sk}_i = (\mathbf{S}_i, \mathbf{V}_i)$, compute partial aggregator key $\mathsf{dk}_i = \mathbf{S}_i + \mathbf{V}_i \pmod q$.

$\mathsf{AggKeygen}(\{\mathsf{dk}_i\}_{i\in[\ell]})$: Given $\{\mathsf{dk}_i\}_{i\in[\ell]}$, compute aggregator decryption key

$$\mathsf{dk}_0 := \sum_{i=1}^{\ell} \mathsf{dk}_i = \sum_{i=1}^{\ell} (\mathbf{S}_i + \mathbf{V}_i) = \sum_{i=1}^{\ell} \mathbf{S}_i \pmod q = \mathbf{S}_0 \tag{6}$$

$\mathsf{Enc}(i, \mathsf{sk}_i, \mathbf{x}_{i,t}, t)$: Given input $\mathbf{x}_{i,t} \in \mathbb{Z}_p^n$ and a timestamp $t = t_j$, generate a vector $\mathbf{t}_j = F_K(j) \in \mathbb{Z}_q^n$. Sample $\mathbf{e}_{i,t} \leftarrow \mathcal{X}^n$ and compute the ciphertext $\mathsf{ct}_{i,t}$ as

$$\mathsf{ct}_{i,t} = \left\lfloor \frac{q}{p} \right\rfloor \cdot \mathbf{x}_{i,t} + \mathbf{t}_j \cdot \mathbf{S}_i^\top + \mathbf{e}_{i,t} \pmod q \tag{7}$$

$\mathsf{AggDec}\big(\mathsf{dk}_0, \{\mathsf{ct}_{i,t}\}_{i\in[\ell]}, t\big)$: Given timestamp $t = t_j$, generate the vector $\mathbf{t}_j = F_K(j) \in \mathbb{Z}_q^n$ and compute the aggregated sum as

$$\mathbf{x}_t = \left[ \left\lfloor \frac{p}{q} \left( \sum_{i=1}^{\ell} \mathsf{ct}_{i,t} - \mathbf{t}_j \cdot \mathbf{S}_0^\top \pmod q \right) \right\rceil \right]_p \tag{8}$$

**Correctness:** Correctness follows similarly as described in Sect. 3.1. At timestamp $t = t_j$, we have

$$\sum_{i=1}^{\ell} \mathsf{ct}_{i,t} - \mathbf{t}_j \cdot \mathbf{S}_0^\top \pmod q = \sum_{i=1}^{\ell} \left\lfloor \frac{q}{p} \right\rfloor \cdot \mathbf{x}_{i,t} + \sum_{i=1}^{\ell} \mathbf{e}_{i,t} \pmod q \tag{9}$$

Observe that for an odd prime $q$,

$$\sum_{i=1}^{\ell} \left\lfloor \frac{q}{p} \right\rfloor \cdot \mathbf{x}_{i,t} = \left\lfloor \frac{q}{p} \right\rfloor \cdot \sum_{i=1}^{\ell} [\mathbf{x}_{i,t}]_p - \frac{1}{2} \left( \sum_{i=1}^{\ell} \mathbf{x}_{i,t} - \sum_{i=1}^{\ell} [\mathbf{x}_{i,t}]_p \right) \tag{10}$$

To make sure that $\left\lfloor \frac{q}{p} \right\rfloor \cdot \sum_{i=1}^{\ell} \mathbf{x}_{i,t} + \sum_{i=1}^{\ell} \mathbf{e}_{i,t}$ does not flow over the modulus $q$, we need to ensure that

$$\left\| \sum_{i=1}^{\ell} \mathbf{e}_{i,t} - \frac{1}{2} \left( \sum_{i=1}^{\ell} \mathbf{x}_{i,t} - \sum_{i=1}^{\ell} [\mathbf{x}_{i,t}]_p \right) \right\|_\infty < \frac{q}{2} \tag{11}$$

This is satisfied when $\frac{\ell}{2}(p + 2B) < \frac{q}{2}$.

**Security:** The security of the above DPSA scheme can be proved using the same proof strategy as described in Sect. 3.2. It can be proved using a hybrid argument consisting of the games outlined in Table 5. Here $\mathsf{G}_0$ corresponds to the $\mathsf{AO}_0$ game where $\mathsf{QChallenge}$ queries are answered with an encryption of $\mathbf{x}_{i,t}^0$

and $\mathsf{G}_3$ corresponds to the $\mathsf{AO}_1$ game where the challenge queries are answered with an encryption of $\mathbf{x}^1_{i,t}$.

The transition from $\mathsf{G}_0$ to $\mathsf{G}_1$ consists of adding perfect secret shares of 0 denoted by $\mathbf{r}_i \leftarrow SS(0)$ to the challenge ciphertexts. It can be achieved by replacing the PRF $F_{\mathbf{S}_i}(\mathbf{t}_j)$ with a random function (RF) and using a sequence of hybrid games as described in Lemma 2. Transition from $\mathsf{G}_1$ to $\mathsf{G}_2$ can be done similarly by changing the PRF with an RF for the two users as described in Case 2 of Lemma 3. Case 1 follows directly from Lemma 3. Finally, transition from $\mathsf{G}_2$ to $\mathsf{G}_3$ consists of making the changes backwards.

**Table 5.** Hybrid games for the AO security of the DPSA scheme in the standard model. Change in each games is highlighted with a square box

| Game | $\mathsf{ct}_{i,t}$ | Justification |
|---|---|---|
| $\mathsf{G}_0$ | $\mathbf{c}_{i,t} \leftarrow F_{\mathbf{S}_i}(\mathbf{t}_j)$<br>$\mathsf{ct}_{i,t} \leftarrow \mathbf{c}_{i,t} + \lfloor q/p \rfloor \cdot \mathbf{x}^0_{i,t}$ | $\mathsf{AO}_0$ game |
| $\mathsf{G}_1$ | $\mathbf{c}'_{i,t} \leftarrow F_{\mathbf{S}_i}(\mathbf{t}_j)$<br>$\boxed{\mathbf{c}_{i,t} \leftarrow \mathbf{c}'_{i,t} + \mathbf{r}_i}, \quad \mathbf{r}_i \leftarrow SS(0)$<br>$\mathsf{ct}_{i,t} \leftarrow \mathbf{c}_{i,t} + \lfloor q/p \rfloor \cdot \mathbf{x}^0_{i,t}$ | $\mathbf{c}_{i,t}$ indistinguishable from random |
| $\mathsf{G}_2$ | $\mathbf{c}'_{i,t} \leftarrow F_{\mathbf{S}_i}(\mathbf{t}_j)$<br>$\mathbf{c}_{i,t} \leftarrow \mathbf{c}'_{i,t} + \mathbf{r}_i, \quad \mathbf{r}_i \leftarrow SS(0)$<br>$\boxed{\mathsf{ct}_{i,t} \leftarrow \mathbf{c}_{i,t} + \lfloor q/p \rfloor \cdot \mathbf{x}^1_{i,t}}$ | information-theoretic |
| $\mathsf{G}_3$ | $\boxed{\mathbf{c}_{i,t} \leftarrow F_{\mathbf{S}_i}(\mathbf{t}_j)}$<br>$\mathsf{ct}_{i,t} \leftarrow \mathbf{c}_{i,t} + \lfloor q/p \rfloor \cdot \mathbf{x}^1_{i,t}$ | $\mathbf{c}_{i,t}$ indistinguishable from random |

## 5 Conclusion

In this paper, we presented a decentralized private stream aggregation (DPSA) protocol that does not rely on a trusted authority for key generation. We gave a formal definition of a DPSA scheme and presented a concrete construction based on the LWE problem both in the random oracle model as well as the standard model. We proved the security of the DPSA scheme under the aggregator obliviousness notion with static corruptions. Further, we discussed possible solutions for practical deployments such as clients joining and leaving the system. In addition, we provided sample parameters for the concrete construction based on the LWE assumption, and demonstrated that our scheme achieves comparable ciphertext sizes to that of SLAP [24] for equivalent plaintext spaces.

**Acknowledgements.** This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

# Appendix

## A Private Stream Aggregation

**Definition 6 (Private Stream Aggregation [21]).** *A private stream aggregation scheme over an input space $\mathcal{M}$ consists of the following PPT algorithms:*

$\mathsf{Setup}(1^\lambda, 1^\ell)$*: Takes as input the security parameter $\lambda$ and number of users $\ell$ and generates public parameters $\mathsf{pp}$, user secret keys $\mathsf{sk}_i$ and aggregator decryption key $\mathsf{dk}_0$. Each user gets the corresponding secret key $\mathsf{sk}_i$ for $i \in [\ell]$ and the aggregator receives the decryption key $\mathsf{dk}_0$. The public parameters $\mathsf{pp}$ is implicitly an input to all the algorithms.*

$\mathsf{Enc}(i, \mathsf{sk}_i, \mathbf{x}_{i,t}, t)$*: Takes as input the user index $i$, the secret key $\mathsf{sk}_i$, the input $\mathbf{x}_{i,t} \in \mathcal{M}$ and generates an encryption of $\mathbf{x}_{i,t}$ using $\mathsf{sk}_i$. Outputs the ciphertext $\mathsf{ct}_{i,t}$.*

$\mathsf{AggDec}(\mathsf{dk}_0, \{\mathsf{ct}_{i,t}\}_{i\in[\ell]}, t)$*: Takes the aggregator decryption key $\mathsf{dk}_0$ and ciphertexts $\{\mathsf{ct}_{i,t}\}_{i\in[\ell]}$ for the time period $t$ and outputs the aggregated sum $\mathbf{x}_t = \sum_{i=1}^{\ell} \mathbf{x}_{i,t}$.*

**Correctness**: The above PSA scheme $\mathsf{PSA} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{AggDec})$ is said to be correct if for any $\lambda, \ell \in \mathbb{N}$, any message $\mathbf{x}_{i,t} \in \mathcal{M}$, it holds that

$$\Pr\left[\mathsf{AggDec}(\mathsf{dk}_0, \{\mathsf{ct}_{i,t}\}_{i\in[\ell]}, t) = \sum_{i=1}^{\ell} \mathbf{x}_{i,t} : \begin{array}{l}(\mathsf{pp}, \{\mathsf{sk}_i\}_{i\in[\ell]}, \mathsf{dk}_0) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell) \\ \mathsf{ct}_{i,t} \leftarrow \mathsf{Enc}(i, \mathsf{sk}_i, \mathbf{x}_{i,t}, t)\end{array}\right] = 1$$

**Definition 7 (Aggregator Obliviousness for PSA).** *The aggregator obliviousness security for a PSA scheme can be defined in terms of the security experiment $\mathsf{AO}_\beta(\lambda, \ell, \mathcal{A})$ given in Fig. 3. No adversary $\mathcal{A}$ should be able to win this game with non-negligible advantage.*

$\underline{\mathsf{AO}_\beta(\lambda, \ell, \mathcal{A})}$

1: $(\mathsf{pp}, \{\mathsf{sk}_i\}_{i\in[\ell]}, \mathsf{dk}_0) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$.
2: $\beta \leftarrow \mathcal{A}^{\mathsf{QCorr}(\cdot),\mathsf{QEnc}(\cdot,\cdot,\cdot,\cdot),\mathsf{QChallenge}(\cdot,\cdot,\cdot,\cdot)}(\mathsf{pp})$
3: **if** condition $(*)$ is satisfied **then**
4: output $\beta$
5: **else**
6: output 0

**Fig. 3.** Aggregator Obliviousness experiment for PSA

*The challenger first runs the Setup algorithm and returns the public parameters $\mathsf{pp}$ to the adversary. The adversary makes queries to the following oracles:*

- ***Corruption oracle*** $\mathsf{QCorr}(i)$*: The adversary submits an integer* $i \in \{1, \ldots, \ell\}$ *and gets back the* $i$*-th user's secret key* $\mathsf{sk}_i$*. If the adversary submits* $i = 0$*, then it gets the aggregator decryption key* $\mathsf{dk}_0$*.*
- ***Encryption oracle*** $\mathsf{QEnc}(i, \mathbf{x}_{i,t}, t)$*: The adversary submits* $(i, \mathbf{x}_{i,t}, t)$ *and receives* $\mathsf{ct}_{i,t}, \leftarrow \mathit{Enc}(i, \mathsf{sk}_i, \mathbf{x}_{i,t}, t)$ *from the challenger.*
- ***Challenge oracle*** $\mathsf{QChallenge}(\mathcal{U}, \{\mathbf{x}^0_{i,t^*}\}_{i\in\mathcal{U}}, \{\mathbf{x}^1_{i,t^*}\}_{i\in\mathcal{U}}, t^*)$*: This query can be made only once by the adversary. The adversary selects a set of users* $\mathcal{U}$ *and time period* $t^*$ *and for each* $i \in \mathcal{U}$*, the adversary chooses two sets of inputs* $\mathbf{x}^0_{i,t^*}$*,* $\mathbf{x}^1_{i,t^*}$*. The challenger randomly samples* $b \leftarrow \{0, 1\}$ *and returns* $\mathsf{ct}_{i,t^*} \leftarrow \mathit{Enc}(\mathsf{sk}_i, \mathbf{x}^0_{i,t^*}, t^*)$ *for all* $i \in \mathcal{U}$ *if* $b = 0$ *and* $\mathsf{ct}_{i,t^*} \leftarrow \mathit{Enc}(\mathsf{sk}_i, \mathbf{x}^1_{i,t^*}, t^*)$ *for all* $i \in \mathcal{U}$ *if* $b = 1$*.*

*Finally, the adversary outputs a guess* $b'$ *for the value of* $b$ *and the experiment outputs* $\beta$ *depending on the following conditions.*

*Let* $\mathcal{CS}$ *be the set of corrupted users,* $\mathcal{HS}$ *be the set of honest users at the end of the game and let* $\mathcal{E}_{t^*}$ *be the set of users for which an encryption query has been made at time* $t^*$*. Let* $\mathcal{Q}_{t^*} := \mathcal{U} \cup \mathcal{E}_{t^*}$ *be the set of users for which* $\mathcal{A}$ *receives an encryption or a challenge ciphertext at timestamp* $t^*$*. The condition* $(*)$ *is satisfied if all of the following conditions hold:*

- $\mathcal{U} \cap \mathcal{CS} = \emptyset$*: The set of users specified during the* Challenge *phase must be uncorrupted at the end of the game.*
- *Adversary* $\mathcal{A}$ *has not queried* $\mathsf{QEnc}(i, \mathbf{x}_{i,t}, t^*)$ *for the same* $i$ *and* $t^*$*. Otherwise, this would violate the encrypt-once policy.*
- $\mathcal{U} \cap \mathcal{E}_{t^*} = \emptyset$*: The adversary cannot query challenge ciphertexts to the users in* $\mathcal{E}_{t^*}$*. In other words, the adversary cannot get a challenge ciphertext from users for which it has queried the encryption oracle at time* $t^*$*.*
- *If the adversary has compromised the aggregator and* $\mathcal{Q}_{t^*} \cup \mathcal{CS} = [\ell]$*, then the following condition must be satisfied.*

$$\sum_{i\in\mathcal{U}} \boldsymbol{x}^0_{i,t^*} = \sum_{i\in\mathcal{U}} \boldsymbol{x}^1_{i,t^*}$$

*We set* $\beta \leftarrow b'$ *if the above conditions are satisfied, otherwise we set* $\beta = 0$*.*

*A PSA scheme is said to be aggregator oblivious if for any PPT adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}$ *such that*

$$\mathsf{Adv}^{\mathsf{AO}}_{\mathcal{A},\mathsf{PSA}}(\lambda, \ell) = |Pr[\mathsf{AO}_0(\lambda, \ell, \mathcal{A}) = 1] - Pr[\mathsf{AO}_1(\lambda, \ell, \mathcal{A}) = 1]| \leq \mathsf{negl}(\lambda)$$

## B Games for the Proof of Theorem 1

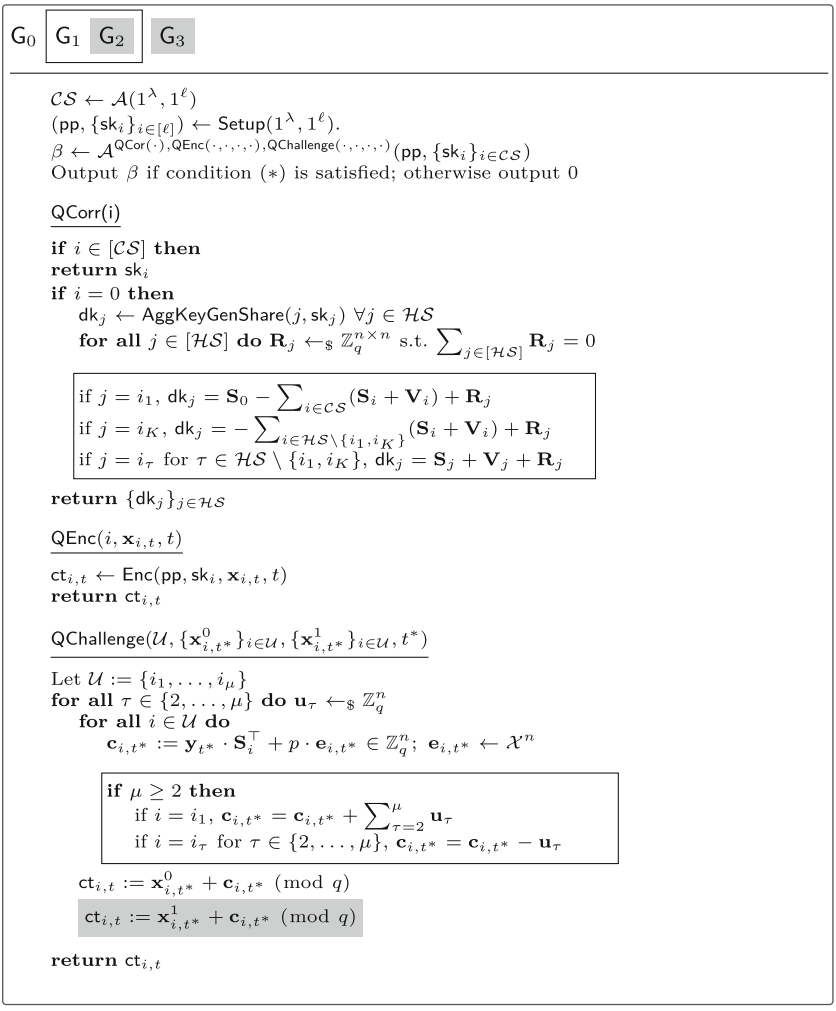


**Fig. 4.** Games for the proof of Theorem 1. Here $\mathcal{HS} := [\ell] \setminus \mathcal{CS}$. Condition $(*)$ is given in Definition 5.

$\mathsf{G}_{0.l-1}$ for $l \in \{1, \ldots, \ell\}$:

$\mathcal{CS} \leftarrow \mathcal{A}(1^\lambda, 1^\ell)$
$i_1^* \leftarrow_\$ [\mathcal{HS}]$, $i_K^* \leftarrow_\$ [\mathcal{HS}] \setminus \{i_1^*\}$
$(\mathsf{pp}, \{\mathsf{sk}_i\}_{i\in[\ell]}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$
$\beta \leftarrow \mathcal{A}^{\mathsf{QCor}(\cdot),\mathsf{QEnc}(\cdot,\cdot,\cdot,\cdot),\mathsf{QChallenge}(\cdot,\cdot,\cdot,\cdot)}(\mathsf{pp}, \{\mathsf{sk}_i\}_{i\in\mathcal{CS}})$
Output $\beta$ if condition $(*)$ is satisfied AND the game was not aborted; otherwise output 0

QCorr(i)

**if** $i \in [\mathcal{CS}]$ **then**
**return** $\mathsf{sk}_i$
**if** $i = 0$ **then**
$\mathsf{dk}_j \leftarrow \mathsf{AggKeyGenShare}(j, \mathsf{sk}_j)\ \forall j \in \mathcal{HS}$
**for all** $j \in [\mathcal{HS}]$ **do** $\mathbf{R}_j \leftarrow_\$ \mathbb{Z}_q^{n\times n}$ s.t. $\sum_{j\in[\mathcal{HS}]} \mathbf{R}_j = 0$

if $j = i_1$, $\mathsf{dk}_j = \mathbf{S}_0 - \sum_{i\in\mathcal{CS}}(\mathbf{S}_i + \mathbf{V}_i) + \mathbf{R}_j$
if $j = i_K$, $\mathsf{dk}_j = -\sum_{i\in\mathcal{HS}\setminus\{i_1,i_K\}}(\mathbf{S}_i + \mathbf{V}_i) + \mathbf{R}_j$
if $j = i_\tau$ for $\tau \in \mathcal{HS} \setminus \{i_1, i_K\}$, $\mathsf{dk}_j = \mathbf{S}_j + \mathbf{V}_j + \mathbf{R}_j$

**return** $\{\mathsf{dk}_j\}_{j\in\mathcal{HS}}$

QEnc$(i, \mathbf{x}_{i,t}, t)$

$\mathbf{c}_{i,t} = \mathbf{y}_t \cdot \mathbf{S}_i^\top + p \cdot \mathbf{e}_{i,t}$
If $i = i_1^*$, $\mathbf{c}_{i,t} = \mathbf{b}_t$, $\mathbf{b}_t \leftarrow_\$ \mathbb{Z}_q^n$
If $i = i_K^*$, $\mathbf{c}_{i,t} = H(t) \cdot \mathbf{S}_0 - H(t) \sum_{j\in[\ell]\setminus\{i_1^*,i_K^*\}} \mathbf{S}_j - \mathbf{b}_t$
$\mathsf{ct}_{i,t} = \mathbf{x}_{i,t} + \mathbf{c}_{i,t}$
**return** $\mathsf{ct}_{i,t}$

QChallenge$(\mathcal{U}, \{\mathbf{x}_{i,t^*}^0\}_{i\in\mathcal{U}}, \{\mathbf{x}_{i,t^*}^1\}_{i\in\mathcal{U}}, t^*)$

Let $\mathcal{U} := \{i_1, \ldots, i_\mu\}$ and $K = min(\mu, l)$
**for all** $\tau \in \{2, \ldots, K\}$ **do** $\mathbf{u}_\tau \leftarrow_\$ \mathbb{Z}_q^n$
**for all** $i \in \mathcal{U}$ **do**
$\mathbf{c}_{i,t^*} := \mathbf{y}_{t^*} \cdot \mathbf{S}_i^\top + p \cdot \mathbf{e}_{i,t^*}$
**if** $K \geq 2$ **then**
if $i_1 \neq i_1^*$ and $i_K \neq i_K^*$ then abort game
if $i = i_1$, $\mathbf{c}_{i,t^*} = \mathbf{c}_{i,t^*} + \sum_{\tau=2}^{K} \mathbf{u}_\tau$
if $i = i_\tau$ for $\tau \in \{2, \ldots, K\}$, $\mathbf{c}_{i,t^*} = \mathbf{c}_{i,t^*} - \mathbf{u}_\tau$
$\mathsf{ct}_{i,t^*} := \mathbf{x}_{i,t^*}^0 + \mathbf{c}_{i,t^*} (\mathrm{mod} q)$
**return** $\mathsf{ct}_{i,t}$

**Fig. 5.** Games for the proof of Lemma 2

## References


1. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. J. Math. Cryptol. **9**(3), 169–203 (2015)
2. Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 595–618. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03356-8_35
3. Banerjee, A., Peikert, C., Rosen, A.: Pseudorandom functions and lattices. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 719–737. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_42
4. Becker, D., Guajardo, J., Zimmermann, K.H.: Revisiting private stream aggregation: lattice-based PSA. In: NDSS 2018. The Internet Society, 2018, vol. 2, p. 5 (2018)
5. Bell, J., et al.: {ACORN}: input validation for secure aggregation. In: 32nd USENIX Security Symposium (USENIX Security 2023), pp. 4805–4822 (2023)
6. Bell, J.H., Bonawitz, K.A., Gascón, A., Lepoint, T., Raykova, M.: Secure single-server aggregation with (poly) logarithmic overhead. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, pp. 1253–1269 (2020)
7. Benhamouda, F., Joye, M., Libert, B.: A new framework for privacy-preserving aggregation of time-series data. ACM Trans. Inf. Syst. Secur. (TISSEC) **18**(3), 1–21 (2016)
8. Bonawitz, K., et al.: Practical secure aggregation for privacy-preserving machine learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 1175–1191 (2017)
9. Brakerski, Z., Langlois, A., Peikert, C., Regev, O., Stehlé, D.: Classical hardness of learning with errors. In: Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing, pp. 575–584 (2013)
10. Brorsson, J., Gunnarsson, M.: Dipsauce: efficient private stream aggregation without trusted parties. IACR Cryptology ePrint Archive (2023). https://eprint.iacr.org/2023/214
11. Chan, T.H.H., Shi, E., Song, D.: Privacy-preserving stream aggregation with fault tolerance. In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 200–214. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32946-3_15
12. Chotard, J., Dufour Sans, E., Gay, R., Phan, D.H., Pointcheval, D.: Decentralized multi-client functional encryption for inner product. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part II. LNCS, vol. 11273, pp. 703–732. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03329-3_24
13. Emura, K.: Privacy-preserving aggregation of time-series data with public verifiability from simple assumptions. In: Pieprzyk, J., Suriadi, S. (eds.) ACISP 2017, Part II. LNCS, vol. 10343, pp. 193–213. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59870-3_11
14. Erkin, Z., Tsudik, G.: Private computation of spatial and temporal power consumption with smart meters. In: Bao, F., Samarati, P., Zhou, J. (eds.) ACNS 2012. LNCS, vol. 7341, pp. 561–577. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31284-7_33
15. Ernst, J., Koch, A.: Private stream aggregation with labels in the standard model. In: Proceedings on Privacy Enhancing Technologies, vol. 2021, no. 4, pp. 117–138 (2021)

16. Fereidooni, H., et al.: Safelearn: secure aggregation for private federated learning. In: 2021 IEEE Security and Privacy Workshops (SPW), pp. 56–62. IEEE (2021)
17. Joye, M., Libert, B.: A scalable scheme for privacy-preserving aggregation of time-series data. In: Sadeghi, A.R. (ed.) FC 2013. LNCS, vol. 7859, pp. 111–125. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39884-1_10
18. Leontiadis, I., Elkhiyaoui, K., Önen, M., Molva, R.: PUDA-privacy and unforgeability for data aggregation. In: Reiter, M., Naccache, D. (eds.) CANS 2015. LNCS, vol. 9476, pp. 3–18. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26823-1_1
19. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: 37th Annual ACM Symposium on Theory of Computing, pp. 84–93 (2005)
20. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM (JACM) **56**(6), 1–40 (2009)
21. Shi, E., Chan, H., Rieffel, E., Chow, R., Song, D.: Privacy-preserving aggregation of time-series data. In: Annual Network & Distributed System Security Symposium (NDSS). Internet Society (2011)
22. Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings 35th Annual Symposium on Foundations of Computer Science, pp. 124–134. IEEE (1994)
23. Takeshita, J., Carmichael, Z., Karl, R., Jung, T.: Terse: tiny encryptions and really speedy execution for post-quantum private stream aggregation. In: Li, F., Liang, K., Lin, Z., Katsikas, S.K. (eds.) SecureComm 2022, pp. 331–352. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-25538-0_18
24. Takeshita, J., Karl, R., Gong, T., Jung, T.: Slap: simpler, improved private stream aggregation from ring learning with errors. J. Cryptol. **36**(2), 8 (2023)
25. Tsaloli, G., Liang, B., Brunetta, C., Banegas, G., Mitrokotsa, A.: Deva: Decentralized, verifiable secure aggregation for privacy-preserving learning. In: Liu, J.K., Katsikas, S., Meng, W., Susilo, W., Intan, R. (eds.) ISC 2021. LNCS, vol. 13118, pp. 296–319. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-91356-4_16
26. Valovich, F.: Aggregation of time-series data under differential privacy. In: Lange, T., Dunkelman, O. (eds.) LATINCRYPT 2017. LNCS, vol. 11368, pp. 249–270. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-25283-0_14
27. Waldner, H., Marc, T., Stopar, M., Abdalla, M.: Private stream aggregation from labeled secret sharing schemes. IACR Cryptology ePrint Archive 2021, 81 (2021). https://eprint.iacr.org/2021/081