Revisiting Private Stream Aggregation: *Lattice-Based* PSA

Daniela Becker
Robert Bosch LLC,
RTC Pittsburgh, USA
daniela.becker2@us.bosch.com

Jorge Guajardo
Robert Bosch LLC,
RTC Pittsburgh, USA
jorge.guajardomerchan@us.bosch.com

Karl-Heinz Zimmermann
Hamburg University of Technology,
Germany
k.zimmermann@tuhh.de

***Abstract*—In this age of massive data gathering for purposes of personalization, targeted ads, etc. there is an increased need for technology that allows for data analysis in a privacy-preserving manner. *Private Stream Aggregation* as introduced by Shi *et al.* (NDSS 2011) allows for the execution of aggregation operations over privacy-critical data from multiple data sources without placing trust in the aggregator and while maintaining differential privacy guarantees. We propose a generic PSA scheme, LaPS, based on the *Learning With Error* problem, which allows for a flexible choice of the utilized privacy-preserving mechanism while maintaining post-quantum security. We overcome the limitations of earlier schemes by relaxing previous assumptions in the security model and provide an efficient and compact scheme with high scalability. Our scheme is practical, for a plaintext space of 2^{16} and 1000 participants we achieve a performance gain in decryption of roughly 150 times compared to previous results in Shi *et al.* (NDSS 2011).**

I. Introduction

Suppose, your local city municipality would like to obtain an overview of the health status of its citizens across different parts of the city. Further suppose, it will outsource the work to a contractor who is responsible for gathering the data from all local hospitals and producing the desired statistics. Clearly, the input data provided by the hospitals is potentially privacy-sensitive as it concerns health-related data. From a standard differential privacy point of view we are concerned with adversaries that are able to derive information from the published statistics about individual data records in the input data (see e.g. [25], [18]). Therefore, we are looking to apply privacy-preserving techniques to the statistics output before publication such that, given the published result, it is hard to derive any privacy-sensitive information about individual data points. In the differential privacy setting, this is achieved by adding *noise* to the raw output, such that the relation between the final output and the input is concealed.

In our example, the contractor will therefore be instructed to add some noise to the produced statistics before outputting it. Figure 1 illustrates this setting: $U_1, \ldots, U_N$ are the hospitals with their respective data $D_1, \ldots, D_N$, which they transmit to the contractor A_t. A_t then evaluates the desired statistical function on the input data and adds some noise N. Finally she publishes the result Σ. Assuming that N is properly distributed, the result Σ is now differentially private.

There are two problems with this approach: first, the hospitals have to send their sensitive data to the contractor. Therefore, the contractor has to be trusted to keep the data confidential and use it only for the original intended purpose. Second, the contractor is also entrusted with adding noise, such that the published output is indeed differentially private. The latter could be partially mitigated by simply funneling back the statistics result to the city municipality and shifting the task of privacy-preserving noise addition and final publication to it. However, this is only a partial solution as the contractor still holds the raw statistics result, he may maliciously take advantage of this information, and full trust still needs to be placed in the municipality.

Therefore, the standard differential privacy model assumes a *trusted* data handler (e.g., the contractor) that executes privacy-preserving operations such that statistics over the users' (e.g. the hospitals) data are differentially private. For the special case of data aggregation the concept of *Private Stream Aggregation* (PSA) eliminates this assumption and incorporates an *untrusted* aggregator. Shi *et al.* [40] were the first to propose a PSA protocol that allows a set of users to compute *sum* aggregates over their data with provable security and privacy guarantees. Their approach tackles the aforementioned issues by designing a scheme that has two core properties: aggregation of *encrypted* data and *distributed* noise generation. Hence, as depicted in Figure 2 each user U_i now adds some noise N_i to their data D_i resulting in a "noisy" value X_i, which she encrypts and transmits to the aggregator A_{sh}. A_{sh} then performs the aggregation over all received ciphertexts and can only decrypt the aggregated result Σ, which she publishes. Note that each plaintext X_i already contains noise. Concretely, Shi *et al.* [40] ensure differential privacy of the output using "randomized" geometric noise, i.e. each user adds geometric noise with a certain probability depending on the number of overall participants. Consequently, the N_i's accumulate into a properly distributed N within the aggregation result Σ such that differential privacy is guaranteed. Hence, in our example the city municipality does not need to rely on the contractor to add noise in order to ensure overall differential privacy and it is not concerned with possible compromise of the patient records either.

Network and Distributed Systems Security (NDSS) Symposium 2018
18-21 February 2018, San Diego, CA, USA
ISBN 1-891562-49-5
http://dx.doi.org/10.14722/ndss.2018.23120
www.ndss-symposium.org

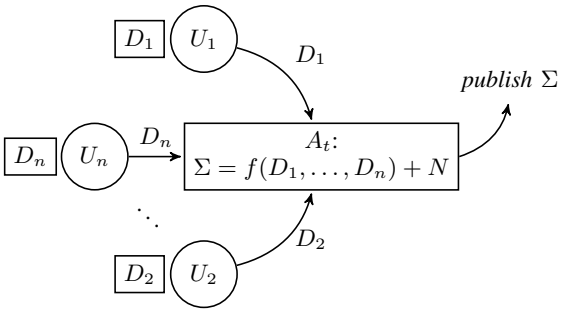

Fig. 1: Standard privacy model: *trusted* aggregator A_t, where $\{U_i\}_{i=1}^n$: users, $\{D_i\}_{i=1}^n$: privacy-sensitive data, respectively

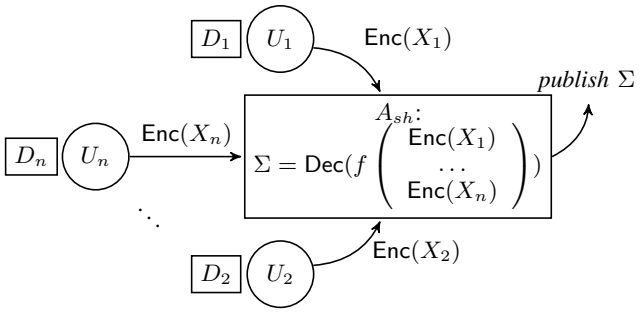

Fig. 2: PSA model: *untrusted* aggregator A_{sh}, where $\{X_i\}_{i=1}^n$ noisy user data

Shi *et al.* [40] left as an open problem the design of a PSA scheme that supports a large plaintext space to compute sums, which is inherently limited in [40] due to their reliance on discrete logarithms. In this work, we solve this challenge by leveraging the *Learning With Errors* (LWE) problem. The LWE problem was introduced in Regev's seminal work [39]. As a novel hardness assumption it lead to a series of different LWE-based cryptosystems and applications, which by extension are also post-quantum secure (e.g. [31], [21], [38], [20], [34]).

A. Our Approach

The *Augmented* LWE (A-LWE) problem introduced by El Bansarkhani *et al.* [16] hides a message $\mathbf{m}$ inside the error-term $\mathbf{e}$ of an otherwise normal looking LWE-term $(\mathbf{A}, \mathbf{b}^T = \mathbf{s}^T\mathbf{A} + \mathbf{e}^T)$. This is possible by first encoding the message into a pseudo-random vector $\mathbf{v}$ and using this vector in order to sample the error term $\mathbf{e} \xleftarrow{\$} D_{\Lambda_{\mathbf{v}}^{\perp}(\mathbf{G}),\alpha q}$. The resulting error distribution is indistinguishable from the standard discrete Gaussian distribution, which is the error distribution regularly used for LWE. The message can be uniquely recovered by retrieving $\mathbf{v} = \mathbf{G}\mathbf{e} \mod q$ and reversing the encoding, where $\mathbf{G}$ is a gadget matrix, i.e. $\mathbf{G} = \mathbf{I} \otimes \mathbf{g}^T$, where $\otimes$ is the Kronecker product, $\mathbf{g}^T = (1, 2, \ldots, 2^{k-1})$ and $q = 2^k$ is the modulus [34] (see Preliminaries in Section III for more details).

As the authors of [16] show, this way of leveraging the error term as a data container yields straightforward encryption capabilities. Note that this is more efficient than typical LWE-based encryption, where a message gets added to an LWE-term and a new LWE-term is needed for each encryption to provide fresh randomness. This A-LWE-based encryption scheme offers higher bandwidth per encryption and fast decryption due to a simple decryption operation. Therefore, using the A-LWE hardness assumption and this particular encryption technique (as opposed to e.g. a basic Regev-style encryption [39] scheme) is instrumental in achieving significant efficiency gains in our resulting PSA scheme.

A solution that does *not* (quite) work. The A-LWE decryption operation is also much more efficient than solving a discrete log. Intuitively, we can construct an A-LWE-based encryption scheme (using for instance the generic scheme provided in [16]) and encrypt each PSA participant's noisy value using this encryption technique rather than the Diffie-Hellman based approach from [40]. Unfortunately, a closer look at the way that the message is encoded in A-LWE reveals that we cannot use A-LWE in its original form in a straightforward manner. The message is XOR-ed with a hash of the secret key: $\mathbf{v} = \mathsf{encode}(H(\mathbf{s}) \oplus \mathbf{m})$. First, the XOR-operation is not additively homomorphic over the integers. Hence, the aggregator would fail to provide an accurate aggregation result. Additionally, note that $\mathbf{s}$ is the individual user's secret key. Therefore, even if we could find a way to ensure the integrity of the sum, correct decryption would require the knowledge of the user's secret keys. This is undesirable as it undermines the idea of *not* trusting the aggregator in the first place.

A solution that *does* work. In order to resolve this dilemma we move away from this "hard-coded" formulation of the encoding step and define a somewhat generalized version of the A-LWE problem, where $\mathbf{v}$ can be generated by *any* additively homomorphic function that takes $\mathbf{m}$ as an input and can be uniquely inverted. Fortunately the authors in [16] show, that from a security perspective the only requirement that $\mathbf{v}$ has to fulfill is indistinguishability from random. Suppose this precondition holds, hardness of the overall A-LWE term holds without any adjustments or additional reductions, i.e. it seamlessly reduces from regular LWE and therefore worst-case lattice problems.

Indistinguishability from random can be achieved by standard crypto techniques. In fact, any additively homomorphic encryption scheme with pseudo-random ciphertexts will satisfy this requirement as long as the addition of ciphertexts corresponds to the addition of plaintexts. In particular, let encryption routine Enc and decryption routine Dec be additively homomorphic. Then $\mathsf{Dec}(\mathsf{Enc}(\sum_i \mathbf{m}_i)) = \mathsf{Dec}(\sum_i \mathsf{Enc}(\mathbf{m}_i))$ for plaintexts $\{\mathbf{m}_i\}$. Additionally, let Enc provide pseudo-random output, then each ciphertext $\mathbf{c}$ is indistinguishable from random. Hence, setting $\mathbf{v} := \mathbf{c}$ immediately fulfills our requirement. If we also assume correct decryption, then we can deterministically recover $\mathbf{m}$ by executing Dec on input $\mathbf{v}$. Viewing the encoding process as an individual building block for the A-LWE-based encryption scheme yields a generic approach to ensuring confidentiality and correct functionality for Private Stream Aggregation.

B. Our Contributions

Summarizing, we define a *generic* A-LWE-based PSA scheme that provides both security and privacy guarantees while allowing for a flexible choice of privacy-preserving noise and where any (additively) homomorphic encryption scheme can be read-

ily deployed in a plug-and-play manner. Our contributions are as follows:

- **Resolve open problem from [40].** We provide a practical solution to the plaintext length restriction that was left as an open problem in [40]. Leveraging the efficiency of the A-LWE hardness assumption, our construction achieves the same differential privacy guarantees as [40] and stronger security guarantees while maintaining high bandwidth efficiency.
- **Post-quantum security and tight reduction.** We achieve provable post-quantum security thanks to the use of schemes based on A-LWE and therefore worst-case lattice problems. Note that since A-LWE reduces seamlessly from LWE, we do not require expensive reductions that would result in large parameters and consequently decreased efficiency.
- **Generic PSA scheme.** We define "LaPS", a generic PSA scheme that allows for a flexible choice of noise distribution to preserve privacy and accounts for future improvements in homomorphic encryption. In particular, our scheme can work with any current or future additively homomorphic semantically secure scheme where the addition of ciphertexts corresponds to the addition of plaintexts[1].
- **No need for encrypt-once model.** We eliminate the "encrypt-once" model required in previous works [40], [42], where part of the scheme's public parameters is used to generate fresh randomness for encryption and it can only be used once in order to ensure secure encryption. Hence, this parameter has to be replaced for each new round of encryption. Our construction inherently guarantees semantic security of the ciphertext and it allows for secure reuse of all public parameters. Note that our scheme can be easily adapted to comply with the encrypt-once paradigm, if desired.
- **Concrete instantiation using BGV and the discrete Laplace mechanism.** We provide a concrete instantiation of our scheme, where we utilize the discrete variant of the commonly utilized Laplace mechanism and a reduced version of the homomorphic encryption scheme BGV adapted from [9].
- **Implementation with improved efficiency.** To the best of our knowledge, we provide the first implementation of a lattice-based PSA scheme. In doing so, our experimental results show high efficiency and compactness. Our reduced version of BGV results in significant performance advantages due to the fact that we do not require homomorphic multiplication in our aggregation scheme. Concretely, the runtime of our decryption routine improves over previous results, i.e. [40], by a factor of 150 and the parameter magnitudes are reduced by several orders of magnitude compared to [9].

[1]Note that several additive homomorphic schemes (e.g., Paillier, Discrete-logarithm as proposed in [40]) do not comply with this requirement and therefore cannot be simply plugged into our construction.

C. Applications

Private-Stream Aggregation has a large set of applications as discussed in [40]. We mentioned the healthcare scenario in the introduction in order to illustrate one such use case but there are many more applications that can benefit from privacy-preserving techniques in order to compute aggregates over sensitive data: for instance *smart meters*, aggregation of *sensor data*, *research surveys*, *cloud services* and *advertising*.

Remark 1: We would like to point out that the idea of basing a PSA scheme on LWE is not new: Valovich [42] introduces an LWE-based PSA scheme, which due to a clever variation of the LWE problem utilizes the noise induced by encryption for privacy purposes. Therefore, in contrast to our scheme, both privacy and security are addressed in one shot. This PSA scheme also allows for a higher plaintext length compared to Shi *et al.* [40]. However, in order to establish hardness of the utilized LWE-variant, one has to resort to a reduction using the lossy code construction from Döttling and Müller-Quade [12], which places significant constraints on the LWE-parameters resulting in a significant decrease in efficiency. While this efficiency gap could be usually compensated by moving into the ring setting (i.e. reducing to ring-LWE instead of regular LWE) this particular variant does not seem to translate into the ring setting as the authors remark [42]. Hence, it is unclear how efficiently this scheme could be implemented in practice. On a more formal note, the aforementioned reduction technique also affects the tightness of the reduction from worst-case lattice problems resulting in a slightly more loose connection to lattice problems. Since we base security of our scheme directly on LWE, our parameters do not suffer from these limitations and our scheme translates into the ring setting in a straightforward manner. As we show in our experimental results we are therefore able to leverage the efficiency boost from the ring-adaptation.

Overview: The remainder of this work is organized as follows. We discuss related work in Section II and introduce notation and definitions in Section III before proposing our generic PSA scheme LaPS in Section IV. We present an example instantiation by first covering the individual building blocks of the scheme in Section V-A and V-B and subsequently describing the details of the complete construction in Section V-C. Section VI evaluates the practical performance of our scheme including runtime results. Finally, we discuss some possible extensions to our scheme in Section VII.

II. Related Work

We have already mentioned the seminal work of [40]. Jung *et al.* [26] also propose a Diffie-Hellman based PSA scheme that is somewhat more general than Shi *et al.*'s [40] as it allows for the evaluation of a general *multi-variate polynomial function*. However the same limitations as [40] also apply here due to the underlying reliance on DDH.

The question of *dynamic joins and leaves*, e.g. due to user failures, was first addressed by Chan *et al.* [8], whose solution we briefly describe in Section VII, but incurs a high aggregation error, which impacts the accuracy of the result. In an effort to minimize the accumulated aggregation error, Li and Cao [29] have each user add very small amounts of noise and order the participants in an interleaved ring structure such that

only a subset of users have to update their cryptographic key when the number of active participants changes due to leaves or joins. However, the resulting communication cost is higher compared to [8] and therefore creates a trade-off between accuracy and communication cost. In contrast, our scheme does **not** incur such a trade-off. Additionally, Li and Cao's [29] scheme (and its extension to computing the minimum in [30]) relies on symmetric-key cryptography.

A. Local Differential Privacy

The concept of *local differential privacy* is a term that is originally found in the data mining and learning theory literature[2], see e.g. [27], and was later formalized in the differential privacy context in [13]. It captures the essence of guaranteeing differential privacy in the distributed setting, i.e. multiple data sources, without the need for a single trusted entity. Instead a differentially private version of the data is created locally at the data owner's end (in our case the user) before any statistical operations are performed. In that sense, PSA schemes, including ours and previous constructions such as [40], provide local differential privacy for the special application case of aggregation. Note that PSA schemes are somewhat special in that they combine differential privacy techniques with cryptographic means like encryption in order to achieve this goal. Therefore, different from other works in the area of local differential privacy, PSA schemes provide both privacy and security guarantees, as we discuss in detail below.

The RAPPOR technique that was first introduced by Erlingsson *et al.* [17] uses *randomized response* techniques in order to provide local differential privacy for *frequency estimation*. More concretely, the goal is to compute the number of occurrences of certain statistical attributes over a population of users, e.g. settings in users' browsers. The attributes, which are composed of fixed strings, are represented by binary vectors, i.e. each element being a predicate indicating the presence or absence of the corresponding attribute, and it is locally encoded using Bloom filters. The resulting output corresponds to the user's randomized response and is differentially private. Each user's response is subsequently shared with a central server, where the desired frequencies over all user strings are decoded using a combination of hypotheses testing, least-squares solving and LASSO regression.

RAPPOR provides privacy through a purely client-based construction and does not require a trusted third party, in contrast to our solution where we assume a trusted setup (see Remark 2). While their construction includes an in-depth formal analysis of the differential privacy guarantees, it is unclear to which extent *security* is guaranteed by the Bloom filters. To date no attack is known against the RAPPOR construction, however a number of attacks on other Bloom-filter based constructions have been proposed in the literature, e.g. an attack on a Bloom-filter based privacy-preserving record linkage system [36] or a denial-of-service attack on network protocols for packet forwarding exploiting information leaked from Bloom filters [1].

[2]Note that earlier related work on randomized response goes back to 1965, i.e. [44], but was not termed as "locally" private then.

Observe that RAPPOR's attack model (see Section 6 in [17]) considers the privacy implications of information that is leaked from a user's report. Our construction eliminates the concern of information leakage from users' outputs as each user data point is *encrypted*. Furthermore, our scheme is equipped with formal security proofs based on known hardness assumptions. In fact, our example instantiation provides a particularly strong security guarantee, namely post-quantum security.

Lastly, although RAPPOR can be extended to numeric user data, it is optimized for categorical input data based on the specific use case of frequency estimation. For instance, randomized response was specifically chosen over other privacy mechanisms, such as the Laplace mechanism, due to this application focus as indicated in [17]. As a result, the input has to be a binary vector representing predicates. Therefore, instantiating RAPPOR for numeric inputs, requires the predicates to be formulated as ranges up to the desired numeric value. For the particular case of computing the sum of user values, a fine-grained distinction would be needed in order to allow for accurate representation of all possible values. Consequently, this may result in a large expansion of the input for larger numbers. In contrast, our construction handles numeric plaintext values in a straightforward way using encryption in addition to allowing for a flexible choice of privacy mechanism.

B. Multi-Party Computation (MPC) and Fully-Homomorphic Encryption (FHE)

MPC is a well-studied field of research [45], [46], [22] (for an overview see e.g. [23]): the idea is to perform computations in a distributed manner, i.e. among N *equal* parties. Informally, every user is supposed to learn the output of the computation but no individual input should be known to anyone but the providing user herself.

For the special application to smart metering, Danezis *et al.* [10] introduce a PSA scheme that extends the aggregation capability to the computation of *non-linear aggregation functions* using secret sharing and MPC-techniques. Their underlying assumption is the existence of a set of trusted authorities that work together in order to compute the desired output while intermediate results remain private. The user distributes shares of her individual value to multiple authorities who can compute linear aggregation functions on their local share before combining them into the output aggregate. The computation of non-linear functions, however, requires interaction among the involved parties that may increase to several rounds of communication with higher complexity of the desired aggregation operation. In contrast, our scheme is non-interactive and does not require an additional set of trusted parties. However, our scheme only supports addition operations.

Different from the PSA case, traditional MPC protocols are initially not designed to have an aggregating party and consequently tackle a conceptually different problem. Using *threshold fully homomorphic encryption* (TFHE) schemes like the one proposed by Asharov *et al.* [3] one may nevertheless apply MPC-protocols to PSA in the following way: in the original TFHE MPC-protocol each of the N users initially

holds a secret and a public key share and after two rounds of key generation the public key shares are combined into a common public encryption key. They rely on a combination of LWE-based encryption and the FHE schemes of Brakerski *et al.* [5], [4] in order to achieve homomorphic operations on the resulting ciphertexts. Hence, by tweaking Asharov *et al.*'s TFHE scheme in order to fit the PSA setting, one would define an aggregator in the following way: one party would simply receive the combined secret key, i.e. the sum of the user's individual secret key shares, which allows for the decryption of any ciphertext that is computed as a function, e.g. the sum, of input ciphertexts from any of the N users.

However, this approach has the following limitations: first, in the PSA setting we require that the aggregator does not learn anything but the aggregate output that is computed as a function of *all* users' inputs [40]. Using the TFHE scheme, the aggregator would be able to decrypt ciphertexts that are built from a subset of the N users' ciphertexts, i.e. also individual ciphertexts, and therefore retrieve individual user's input values. Furthermore, the encryption scheme in [3] is based on Regev's construction [39], which only allows for the encryption of a single bit. Hence, even if the TFHE setting is redefined to satisfy the security requirements of a regular PSA scheme, the plaintext space is still very limited. In contrast, our scheme leverages an A-LWE-based encryption scheme, which extends the available plaintext space and allows for significant efficiency improvements while guaranteeing security in the PSA sense.

C. Augmented Learning With Errors

Regev [39] introduced the *Learning With Error* (LWE) problem, which is appreciated both for its mathematical simplicity and powerful connection to lattice problems. Regev showed that LWE is as hard as certain lattice problems in the worst case, therefore it is considered post-quantum secure. As a novel hardness assumption it has served as the basis for a myriad of new cryptographic constructions: from encryption schemes [39], [32], [5], [31], [33] over key-exchange protocols [28], [11] to oblivious transfer techniques [38].

In order to realize the security guarantees of a PSA scheme the utilization of encryption is essential. As mentioned previously LWE has a number of advantages over other hardness assumptions in this context. However, we would like to point out that there exist several LWE-*variants*, whose choice impacts the practicality of the resulting PSA-scheme: for instance using regular LWE as defined in [39] together with Regev-style encryption would also yield post-quantum secure PSA but would restrict the plaintext to one bit. The corresponding multi-bit version [38] and other LWE-based constructions (e.g. [31]) are also limited in terms of efficiency. Observe that most LWE-based encryption schemes use a one-time-pad-like technique for encryption, where LWE-equations constitute the pad. El Bansarkhani *et al.* [16] were the first to leverage LWE's error-term in order to hide the message and therefore propose a conceptually new way of encryption. Hence, the plaintext size can be significantly increased, which yields critical improvement in bandwidth efficiency. The decryption engine within the resulting encryption scheme is also particularly efficient compared to previous LWE-based schemes [16]. Furthermore, the problem is as hard as LWE without major parameter restrictions and the reduction from worst-case lattice problems remains as tight as in [39]. Hence, there is no security-efficiency trade-off.

Therefore, although LaPS could have been defined using for instance Regev's LWE-based encryption scheme, we chose El Bansarkhani *et al.*'s [16] LWE-version, namely A-LWE, and the corresponding encryption scheme in order to optimize bandwidth and overall efficiency of the resulting PSA scheme.

Lyubashevksy *et al.*'s [32] introduction of the *Ring*-LWE problem, which is the counterpart to Regev's LWE in the ring setting, created an especially compact construction based on the hardness of worst-case problems on *ideal* lattices. Consequently, it is generally considered the most efficient variant of LWE and most constructions are implemented in the ring setting in practice in order to leverage this efficiency advantage in practice. Following standard practice in the literature around LWE (see e.g. [31], [34]) we also first present our generic PSA scheme based on the generalized A-LWE hardness assumption and instantiate a concrete example in the ring setting, i.e. based on *Ring*-A-LWE, which we also use for our implementation.

III. Preliminaries

A. Notation

We denote vectors and matrices using bold lower-case and upper-case letters, respectively. We denote drawing a variable v uniformly at random from some distribution D by $v \xleftarrow{\$} D$. $\log$ denotes the logarithm base 2 and $\ln$ denotes the natural logarithm. The exponential function is denoted by $\exp$. $\otimes$ denotes the Kronecker product of two matrices. The l_1-norm of a database D is denoted $\|D\|_1$ and is defined as $\|D\|_1 = \sum_{i=1}^{|\mathcal{D}|} |D_i|$, where $\mathcal{D}$ denotes the universe of records according to [14]. Note that we abbreviate "generalized" by "gen." in our definitions.

B. Differential Privacy

We first restate a few standard notions from the differential privacy literature before providing more notions that we will use in our construction.

Definition 1 (Distance between databases [14]): The l_1 distance between two databases D_0 and D_1 is $\|D_0 - D_1\|_1$. Two databases D_0 and D_1 are *adjacent* if and only if $\|D_0 - D_1\|_1 \leq 1$.

Definition 2 (Differential Privacy (DP)[14]): A randomized algorithm $\mathcal{M}$ with domain $\mathcal{D}^n$ and range $\mathcal{R}^k$ is (ϵ, δ)-differentially private if for all adjacent databases D_0, D_1 and for all $R \subseteq \mathcal{R}^k$: $\Pr[\mathcal{M}(D_0) \in R] \leq \exp(\epsilon) \Pr[\mathcal{M}(D_1) \in R] + \delta$, where the probability space is over the coin flips of the mechanism $\mathcal{M}$. If $\delta = 0$, we say that $\mathcal{M}$ is ϵ-DP.

Definition 3 (Accuracy [41]): The output of a mechanism $\mathcal{M}$ achieves (α, β)-accuracy for a query $f : \mathcal{D}^n \to \mathcal{R}^k$ if for all $D \in \mathcal{D}^n$: $\Pr[|\mathcal{M}(D) - f(D)| \leq \alpha] \geq 1 - \beta$. The probability space is defined over the randomness of $\mathcal{M}$.

We use the discretized version of the common Laplace mechanism in order to preserve privacy. We restate the definition of the underlying discrete distribution below.

Definition 4 (Discrete Laplace (DLap) [24]): The discrete Laplace (DLap) distribution with scale $\varsigma > 1$ and parameter $p = \exp(-1/\varsigma) \in (0,1)$ is the distribution supported on $\mathbb{Z}$ with probability mass function

$$DLap_{\varsigma}(x) = \frac{1-p}{1+p}p^{|x|} = \frac{1-\exp(-\frac{1}{\varsigma})}{1+\exp(-\frac{1}{\varsigma})}\exp\Big(-\frac{|x|}{\varsigma}\Big).$$

C. Private Stream Aggregation

We start by recalling the main Private Stream Aggregation definition by Shi *et al.* [40].

Definition 5 (Private Stream Aggregation (PSA) Scheme [40]): Let $[n] := \{1, 2, \ldots, n\}$ be the set of users participating in the aggregation each holding values from some domain $\mathcal{D}$. Let $f : \mathcal{D}^n \to \mathcal{R}$ be an aggregation function with some range $\mathcal{R}$. Let $\chi : \mathcal{D} \times \Omega \to \mathcal{D}$ denote some randomization function that adds the two input values from $\mathcal{D}$ and Ω, where Ω is some sample space of the randomization noise. Let T be the set of time steps used throughout execution. A PSA scheme consists of the following PPT-algorithms:

- $(\mathsf{param}, \{sk_i\}, sk_A) \leftarrow \mathsf{Setup}(1^\kappa)$: Takes in a security parameter κ and outputs public parameters param, a private key sk_i for each participant, as well as an aggregator capability sk_A needed for decryption of aggregate statistics in each time step $t \in T$. Each participant i obtains the private key sk_i and the data aggregator obtains the capability sk_A.
- $c_{i,t} \leftarrow \mathsf{NoisyEnc}_i(\mathsf{param}, sk_i, t, d, r)$: During time step t, each participant calls the $\mathsf{NoisyEnc}_i$ algorithm to encode its data d with noise r. The result is a noisy encryption c of d randomized with the noise r.
- $f(\mathbf{x}) \leftarrow \mathsf{AggrDec}(\mathsf{param}, sk_A, t, c_{1,t}, c_{2,t}, \ldots, c_{n,t})$: The decryption algorithm takes in the public parameters param, a capability sk_A, and ciphertexts $c_{1,t}, c_{2,t}, \ldots, c_{n,t}$ for the same time step t. For each $i \in [n]$, let $c_{i,t} = \mathsf{NoisyEnc}_i(sk_i, t, x_i)$, where each $x_i := \chi(d_i, r_i)$ for some randomization function χ. Let $\mathbf{d} := (d_1, \ldots, d_n)$ and $\mathbf{x} = (x_1, \ldots, x_n)$. The decryption algorithm outputs $f(\mathbf{x})$ which is a noisy version of the targeted statistics $f(\mathbf{d})$.

In our definitions we sometimes abuse notation and only require the user to input her raw value d to $\mathsf{NoisyEnc}$ (as opposed to both d and r), when noise r is generated within the routine. Furthermore, we simplified notation by omitting the time-step t in indices. The original notation from Definition 5 is mainly useful in adopting the corresponding security notion of *aggregator obliviousness*, which was also introduced in [40].

Definition 6 (Aggregator Obliviousness [40]): A PSA scheme is aggregator oblivious if no PPT adversary has more than negligible advantage in κ in winning the following security game:

Setup. Challenger runs the Setup algorithm, returns the public parameters param to the adversary.

Queries. The adversary makes the following types of queries adaptively.

- **Encrypt.** The adversary may specify (i, d, r) and ask for the ciphertext. The challenger returns the ciphertext $\mathsf{NoisyEnc}_i(\mathsf{param}, sk_i, t, d, r)$ to the adversary.
- **Compromise.** The adversary specifies an integer $i \in \{0, \ldots, n\}$. If $i = 0$, the challenger returns the aggregator's decryption key sk_A to the adversary. If $i \neq 0$, the challenger returns sk_i, the secret key of the i^{th} participant, to the adversary.
- **Challenge.** This query can only be made once throughout the game. The adversary specifies a set of participants U and a time t, such that $i \in U$ has not been previously compromised. For each user $i \in U$ the adversary chooses two plaintext-noise pairs (d_i, r_i) and (d'_i, r'_i) and sends them to the challenger. The challenger flips a random bit b. If $b = 0$, the challenger computes $\forall i \in U : c_i = \mathsf{NoisyEnc}_i(\mathsf{param}, sk_i, t, d_i, r_i)$. If $b = 1$, $\forall i \in U : c_i = \mathsf{NoisyEnc}_i(\mathsf{param}, sk_i, t, d'_i, r'_i)$ and returns $\{c_i\}$ to the adversary.

Guess. The adversary guesses, whether b is 0 or 1.

We say that the adversary wins the game if she correctly guesses b and if she compromised the aggregator (i.e. possesses the decryption key sk_A,) then $\sum_{i \in U} d_i + r_i = \sum_{i \in U} d'_i + r'_i$ must hold.

Note that if the aggregator colludes with a subset of the participants or is leaked some of the plaintexts[3], then he can inevitably learn the sum of the remaining participants' values. We require that in this case the aggregator learns no *additional* information about these participants' data. However, this requirement is achieved by the *privacy* guarantees of the scheme.

D. LWE and Gaussian Distribution

A *Learning With Error* (LWE) term consists of the coefficient matrix $\mathbf{A}$ and the vector $\mathbf{b} = \mathbf{s}^T\mathbf{A} + \mathbf{e}^T$, where the elements of $\mathbf{A} \in \mathbb{Z}_q^{\kappa \times \lambda}$ and the secret $\mathbf{s} \in \mathbb{Z}_q^\kappa$ are sampled uniformly at random from the q-ary field. The error term $\mathbf{e} \in \mathbb{Z}_q^\lambda$ is sampled according to a discretized Gaussian distribution. Given only $(\mathbf{A}, \mathbf{b})$ it is hard to recover $\mathbf{s}$ or $\mathbf{e}$. In fact, the hardness of the LWE problem can be reduced from worst-case lattice problems, even for the decision LWE problem [37].

After the introduction of the LWE problem in [39] a multitude of LWE-based cryptographic applications have emerged. The need for more efficient cryptographic schemes lead to the definition of several LWE-variants, which would allow for more practical implementation. While most of these variants typically come at the cost of some security relaxation, El Bansarkhani *et al.* [16] presented Augmented LWE (A-LWE), which leverages the error term as a data container and hence provides for very efficient and compact encryption schemes.

An A-LWE term $(\mathbf{A}, \mathbf{b}^T)$ consists of the matrix $\mathbf{A}$ and the vector $\mathbf{b}^T = \mathbf{s}^T\mathbf{A} + \mathbf{e}^T$, where $\mathbf{s}$ is the secret key and $\mathbf{e}$ is the error term. $\mathbf{e}$ is sampled from the special Gaussian distribution

[3]This applies analogously to the adversary who compromises all the secret keys (including the aggregator's) but one.

$D_{\Lambda^{\perp}_{\mathbf{v}}(\mathbf{G}),r}$ that is indistinguishable from the regular discrete Gaussian distribution used for standard LWE. The relation between $\mathbf{e}$ and $\mathbf{v}$ is: $\mathbf{Ge} \equiv \mathbf{v} \mod q$. Hence, given $\mathbf{G}$ and $\mathbf{e}$, one can efficiently recover $\mathbf{v}$. This is essential to the idea of *message embedding* that underlies the A-LWE concept, since a message m is "baked into" $\mathbf{v}$ by producing $\mathbf{v}$ as an encoding of m, such that $\mathbf{v}$ is indistinguishable from random.

We define a slightly more general variant of the A-LWE problem than the original definition from El Bansarkhani *et al.* [16], since we allow for *any* function f embedding the message m such that $\mathbf{v} = f(m)$ as long as output $\mathbf{v}$ is indistinguishable from random rather than requiring the particular way $\mathbf{v}$ is generated in [16]. Note that in the following we refer to the Gaussian parameter αq as σ for better readability. We present the resulting definition below.

Definition 7 (Gen. A-LWE *Distr. (adapt. from [16])):* Let κ, λ, q, l, x be integers, where $l = \lceil \log q \rceil$ and $\lambda = x \cdot l$. Let f be some function where the output is indistinguishable from random. Let $\mathbf{g}^T = (1, 2, \ldots, 2^{l-1}) \in \mathbb{Z}_q^l$ and $\mathbf{G} = \mathbf{I}_{\lambda/l} \otimes \mathbf{g}^T \in \mathbb{Z}_q^{\lambda/l \times \lambda}$. For $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^{\kappa}$ and $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{\kappa \times \lambda}$, define the A-LWE distribution $L^{\mathsf{A\text{-}LWE}}_{\kappa,\lambda,\sigma}(m)$ with $m \in \mathbb{Z}_q$ to be the distribution over $\mathbb{Z}_q^{\kappa \times \lambda} \times \mathbb{Z}_q^{\lambda}$ obtained as follows:

- Set $\mathbf{v} = f(m) \in \mathbb{Z}_q^{\lambda/l}$.
- Sample $\mathbf{e} \leftarrow D_{\Lambda^{\perp}_{\mathbf{v}}(\mathbf{G}),\sigma} \in \mathbb{Z}_q^{\lambda}$.
- Return $(\mathbf{A}, \mathbf{b}^T)$ where $\mathbf{b}^T = \mathbf{s}^T\mathbf{A} + \mathbf{e}^T$.

Definition 8 (Gen. decision A-LWE *(adapt. from [16])):* Let κ, λ, q be integers. Let f be some function with pseudo-random output. The decision $\mathsf{A\text{-}LWE}^f_{\kappa,\lambda,\sigma}$ problem asks to distinguish in polynomial time (in κ) between samples $(\mathbf{A}_i, \mathbf{b}_i^T) \leftarrow L^{\mathsf{A\text{-}LWE}}_{\kappa,\lambda,\sigma}(m)$ and uniform random samples from $\mathbb{Z}_q^{\kappa \times \lambda} \times \mathbb{Z}_q^{\lambda}$ for a secret $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^{\kappa}$ and some $m \in \mathbb{Z}_q$.
We say that decision $\mathsf{A\text{-}LWE}^f_{\kappa,\lambda,\sigma}$ is hard if all polynomial time algorithms solve the decision $\mathsf{A\text{-}LWE}^f_{\kappa,\lambda,\sigma}$ problem only with negligible probability.

In order to prove that security of the resulting ciphertext still holds, we will need Lemma 1, which states that the distribution of the A-LWE error term is indistinguishable from the discrete Gaussian distribution under certain conditions.

Lemma 1 ([16]): Let $\mathbf{M} \in \mathbb{Z}_q^{a \times b}$ be an arbitrary full-rank matrix. If the distribution of $\mathbf{v} \in \mathbb{Z}_q^a$ is computationally indistinguishable from the uniform distribution over $\mathbb{Z}_q^a$, then $D_{\Lambda^{\perp}_{\mathbf{v}}(\mathbf{M}),r}$ is computationally indistinguishable from $D_{\mathbb{Z}^b,r}$ for $r \geq \eta_{\epsilon}(\Lambda^{\perp}(\mathbf{M}))$.

Finally, we utilize the following facts about Ring-LWE for our example instantiation.

*Definition 9 (Ring-*LWE *[6]):* For all $\kappa \in \mathbb{N}$, let $f(x) = f_{\kappa}(x) \in \mathbb{Z}[X]$ be a polynomial of degree $n = n(\kappa)$, let $q = q(\kappa) \in \mathbb{Z}$ be a prime integer, let $t = t(\kappa) \in \mathbb{Z}_q^*$ (thus t and q are relatively prime), let the ring $R = \mathbb{Z}[X]/\langle f(X) \rangle$ and $R_q = r/qR$, and let χ denote a distribution over the ring R.
The decision Ring-LWE problem asks to distinguish in polynomial time (in κ) between any $l = \mathsf{poly}(\kappa)$ samples $(a_i, a_i \cdot s + t \cdot e_i)_{i \in [l]}$ and l uniform random samples from $R_q \times R_q$, where s is sampled from the error distribution χ, a_i are uniform in R_q and the error polynomials e_i are sampled from the error distribution χ.

IV. Generic PSA-Scheme

Prior work on PSA-schemes considers the use of homomorphic encryption problematic, since it is usually designed for a single decryption key and thus the aggregator does not only learn the aggregate but also the individual users' values, which is clearly undesirable. We circumvent this dilemma by *embedding* a homomorphic encryption scheme inside the A-LWE-based PSA scheme. At a high level the user's plaintext is encrypted into an additively homomorphic ciphertext and it is subsequently wrapped into an A-LWE term, which yields the final ciphertext released to the aggregator. Receiving all ciphertexts the aggregator's first decryption key is designed in such a way that it only allows for correct decryption of the sum of the ciphertexts. Using this key, she can successfully lift the outer "layer of encryption", i.e. unveil the inner homomorphic ciphertext. Her second decryption key then allows for decryption and recovery of the summed plaintext. Note that the aggregator still learns nothing more than the noisy sum: since she can only lift the first layer by summing the ciphertexts together, the "inner" ciphertexts are also summed and the recovered plaintext is automatically summed as well.

Zooming in on the technical details, a ciphertext in our scheme is an A-LWE term $(\mathbf{A}, \mathbf{b}^T)$ as presented in Definition 8: it consists of the matrix $\mathbf{A}$, which in our case is distributed as part of the public parameters, and the vector $\mathbf{b}^T = \mathbf{s}^T\mathbf{A} + \mathbf{e}^T$, with $\mathbf{s}$ being the secret key and $\mathbf{e}$ the error term. Recall that $\mathbf{e}$ is sampled from distribution $D_{\Lambda^{\perp}_{\mathbf{v}}(\mathbf{G}),r}$ where $\mathbf{v}$ is the encoding of the plaintext message m. We can recover the message using the gadget matrix $\mathbf{G}$, which is also part of the public parameters in our scheme, using the relation $\mathbf{Ge} \equiv \mathbf{v} \mod q$.

Our generalized version of the A-LWE problem according to Definition 8 allows for any encoding function as long as the output $\mathbf{v}$ is indistinguishable from random. Requiring that the function be additively homomorphic guarantees the correct format for our PSA scheme and regarding it as the encryption routine $\mathsf{AHOM.Enc}$ of a scheme $\mathsf{AHOM} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with pseudo-random ciphertexts ensures pseudo-randomness of the output $\mathbf{v}$.

As a result we get the "layering" approach described above: after adding privacy-preserving noise r_i to user i's database value d_i, i.e. $x_i = d_i + r_i \mod q$, x_i is encrypted using the additively homomorphic routine $\mathsf{AHOM.Enc}$ (under the public key pk generated from $\mathsf{AHOM.Gen}$), which outputs the ciphertext $\mathbf{v}_i$. $\mathbf{v}_i$ is then used to sample $\mathbf{e}_i$, which finally is wrapped into the A-LWE term $\mathbf{c}_{i,\mathbf{A}} = \mathbf{s}_i^T\mathbf{A} + \mathbf{e}_i$. $\mathbf{c}_{i,\mathbf{A}}$ is the output ciphertext.

The definition of the aggregator's decryption key sk_{A_1} ensures that the process can only be reversed for the sum of the ciphertexts from *all* participants. Recall that the secret $\mathbf{s}_i$ within each ciphertext $\mathbf{c}_{i,\mathbf{A}}$ is the user's individual secret encryption key and unknown to the aggregator. Therefore, summing N ciphertexts yields a new ciphertext whose secret is the sum of all N secrets. Therefore, sk_{A_1} is defined as the negative sum over all secrets, such that adding $sk_{A_1}^T\mathbf{A}$ to the new ciphertexts lifts the "secret mask" and recovers the sum of

the error terms $\mathbf{e}_i$. In order to uncover the sum of the $\mathbf{v}_i$'s, i.e. the inner ciphertexts, we multiply by matrix $\mathbf{G}$ utilizing the special property of the error term in A-LWE. In a last step the sum of the $\mathbf{x}_i$'s, i.e. the encrypted messages, is "de-packaged" by invoking AHOM.Dec with the second decryption key sk_{A_2} that was generated by AHOM.Gen in the Setup phase.

Note that El Bansarkhani *et al.*'s generic encryption scheme relies on a trapdoor construction in order to retrieve both error and secret from a given A-LWE-term in the decryption process. As the authors note, this is the most computationally intensive operation of the scheme, therefore the overall efficiency corresponds to the efficiency of executing the trapdoor inversion. In our construction we do not need to rely on expensive trapdoor algorithms since the secret mask is lifted with a single addition due to the nature (inverse of sum of secrets) of the aggregator's secret decryption key.

In the following we first present the formal definition of our generic A-LWE-based PSA Scheme LaPS and then analyze correctness, security and privacy properties.

Algorithm 1 Algorithm Sample to sample from $\Lambda^{\perp}(\mathbf{g}^T)$ [16]

Input: $\mathbf{g}^T \in \mathbb{Z}_q^l, w \in \mathbb{Z}_q, r$
Output: $\mathbf{t} = (t_0, \ldots, t_{l-1})^T \in \Lambda_w^{\perp}(\mathbf{g}^T)$ distributed according to $D_{\Lambda_w^{\perp}(\mathbf{g}^T),r}$
$a_0 := w$
for $j = 0, \ldots, l-1$ **do**
  $t_j \leftarrow D_{2\mathbb{Z}+a_j,r}$
  $a_{j+1} = (a_j - t_j)/2$
end for

Definition 10 (Lattice-based PSA (LaPS)): Let κ be a security parameter, $N \in \mathbb{N}$ the number of participants and let $\alpha \in (0,1)$ and $\beta \in (0,1]$. Let χ be a discrete noise distribution. Let $\mathsf{AHOM} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an asymmetric encryption scheme with pseudo-random ciphertexts that is additively homomorphic, i.e. $\mathsf{AHOM.Dec}(sk, \sum_{i=1}^{N} \mathsf{AHOM.Enc}(pk, m_i)) = \mathsf{AHOM.Dec}(sk, \mathsf{AHOM.Enc}(pk, \sum_{i=1}^{N} m_i))$.
A post-quantum secure Private Stream Aggregation scheme $PQ\text{-}SPA = (\mathsf{Setup}, \mathsf{NoisyEnc}, \mathsf{AggrDec})$ consists of the following PPT-algorithms:

- $(\{\mathbf{A}, \mathbf{g}^T, pk\}, \{s_i\}, (sk_{A_1}, sk_{A_2})) \leftarrow \mathsf{Setup}(1^{\kappa})$: Generate the public parameters $\mathbf{A}$, $\mathbf{g}^T$ and pk as follows and distribute them to all parties.
  - Draw $\mathbf{A}$ uniformly at random from $\mathbb{Z}_q^{\kappa \times \lambda}$, where $l = \lceil \log q \rceil$ and $\lambda = x \cdot l$ for some positive integer x.
  - Set vector $\mathbf{g}^T = (1, 2, \ldots, 2^{l-1}) \in \mathbb{Z}_q^l$.
  - Generate $(pk, sk) \leftarrow \mathsf{AHOM.Gen}$ and extract public key $pk \in (pk, sk)$.

  For all $i \in \{1, \ldots, N\}$ draw $\mathbf{s}_i \leftarrow \mathbb{Z}_q^{\kappa}$ and send it to user i as her secret key.
  The aggregator's secret decryption key is the tuple (sk_{A_1}, sk_{A_2}), where
  - $sk_{A_1} = -\sum_{i=1}^{N} \mathbf{s}_i$ and
  - $sk_{A_2} = sk \in (pk, sk) \leftarrow \mathsf{AHOM.Gen}$.
- $\mathbf{c}_{i,\mathbf{A}} \leftarrow \mathsf{NoisyEnc}_i(\{\mathbf{A}, \mathbf{g}^T, pk\}, \mathbf{s}_i, d_i)$: Each user i takes her data $d_{i,\mathbf{A}} \in \mathcal{D}$ and adds some noise r_i to it, s.t. $x_i = d_i + r_i \mod q \in \mathbb{Z}_q$.
  - r_i is sampled as follows: $r_i = \begin{cases} 0 & \text{with probability } 1-\beta \\ Y & \text{with probability } \beta \end{cases}$, where $Y \leftarrow \chi$.
  - Compute $\mathbf{v}_i = \mathsf{AHOM.Enc}(pk, x_i) \in \mathbb{Z}_q^{\lambda/l}$.
  - Invoke Algorithm 1 for each component of $\mathbf{v}_i$ in order to sample $\mathbf{e}_i \leftarrow D_{\Lambda_{\mathbf{v}_i}^{\perp}(\mathbf{G}),\sigma} \in \mathbb{Z}_q^{\lambda}$, i.e. $\mathbf{e}_i = (\mathsf{Sample}(\mathbf{g}^T, v_{i_1}, \sigma), \ldots, \mathsf{Sample}(\mathbf{g}^T, v_{i_{\lambda/l}}, \sigma))$.

  Output the ciphertext $\mathbf{c}_{i,\mathbf{A}} = \mathbf{s}_i^T \mathbf{A} + \mathbf{e}_i^T \in \mathbb{Z}_q^{\lambda}$.
- $\sum_{i=1}^{N} x_i \leftarrow \mathsf{AggrDec}(\{\mathbf{A}, \mathbf{g}^T\}, (sk_{A_1}, sk_{A_2}), \{\mathbf{c}_{1,\mathbf{A}}, \ldots, \mathbf{c}_{N,\mathbf{A}}\})$: Receiving the users' ciphertexts $\{\mathbf{c}_i\}$ the aggregator computes $\mathbf{c} = \sum_{i=1}^{N} \mathbf{c}_{i,\mathbf{A}}$.
  - Compute $\mathbf{e} = \sum_{i=1}^{N} \mathbf{e}_i^T = \mathbf{c} + sk_{A_1}^T \mathbf{A}$.

  The aggregator retrieves the noisy sum of the users' values via

$$\sum_{i=1}^{N} x_i = \mathsf{AHOM.Dec}(sk_{A_2}, \mathbf{G} \cdot \mathbf{e} \mod q),$$

  where $\mathbf{G} = \mathbf{I}_{\lambda/l} \otimes \mathbf{g}^T \in \mathbb{Z}_q^{\lambda/l \times \lambda}$.

Note that in previous work ([40], [42]) the users' communication was partitioned into individual time-steps t (see Definition 5). Hence, each round of ciphertexts to be aggregated would be associated with a new time-share. While for instance Shi *et al.* imposes this *encrypt-once* model for security reasons, since each new time-step is used to generate fresh randomness for encryption, our scheme is not dependent on separating each round of communication. Therefore we use a single public matrix $\mathbf{A}$ that can be reused throughout the scheme. However, should an application require identifiable time-steps, our scheme can be extended in a straightforward way by generating a set of matrices $(\mathbf{A}_1 \xleftarrow{\$} \mathbb{Z}_q^{\kappa \times \lambda}, \ldots, \mathbf{A}_t \xleftarrow{\$} \mathbb{Z}_q^{\kappa \times \lambda})$ where t is the number of time-steps during Setup and distributing these to each user. Note that this can be implemented in a memory-conserving manner by only storing a seed that is sufficient to generate each matrix $\mathbf{A}_i$ on-the-fly. All security definitions and proofs throughout this work can be extended, accordingly.

We give general security and privacy guarantees with only minimal assumptions on the respective properties of the used building blocks. Concretely, security of the overall PSA scheme requires semantic security with pseudo-random ciphertexts of the embedded homomorphic encryption scheme and (ϵ, δ)-differential privacy of the final output assumes ϵ-differential privacy of the utilized privacy mechanism.

Correctness: Note that $\mathbf{G} \cdot \mathbf{e}_i \mod q = \mathbf{v}_i$, where $\mathbf{e}_i \leftarrow D_{\Lambda_{\mathbf{v}_i}^{\perp}(\mathbf{G}),\sigma}$. Therefore, the input to AHOM.Dec in the AggrDec routine is $\sum_{i=1}^{N} \mathbf{v}_i$. Hence, as long as $\mathsf{AHOM.Dec}(\sum_{i=1}^{N} \mathbf{v}_i) = \sum_{i=1}^{N} x_i$, the aggregator indeed outputs the noisy sum aggregate of the users' values. This is the case by construction, since we require AHOM to be additively homomorphic and therefore the sum of the homomorphic ciphertexts $\mathbf{v} = \sum_{i=1}^{N} \mathbf{v}_i$ corresponds to an encryption of the sum of the underlying plaintexts x_i.

Security: We state security of our scheme as follows:

Theorem 1 (Semantic Security): Let the output of AHOM.Enc be indistinguishable from random.

Then, the ciphertexts generated by NoisyEnc in LaPS according to Definition 10 are semantically secure for $\sigma \geq 2\sqrt{\kappa} \geq 2 \cdot \sqrt{\ln(2n(1+1/\epsilon))/\pi}$ assuming the hardness of worst case lattice problems.

Proof: First, note that due to the above assumption, $\mathbf{v}_i$ is indistinguishable from random. Furthermore, the smoothing parameter $\eta_\epsilon(\Lambda_q^\perp(\mathbf{G}))$ can be bounded from above using[4] Lemma 2.6 in [20] resulting in $\eta_\epsilon(\Lambda_q^\perp(\mathbf{G})) \leq 2 \cdot \sqrt{\ln(2n(1+1/\epsilon))/\pi}$. Therefore, by construction $\sigma \geq \eta_\epsilon(\Lambda_q^\perp(\mathbf{G}))$ and Lemma 1 can be applied to matrix $\mathbf{G}$. Then, $D_{\Lambda_{\mathbf{v}_i}^\perp(\mathbf{G}),\sigma}$ correctly simulates the discrete Gaussian distribution $D_{\mathbb{Z}^\lambda,\sigma}$ and the ciphertexts $\mathbf{c}_i$ represent plain A-LWE$_{\kappa,\lambda,\sigma}$ samples. Note that in the remainder of this section we assume that σ and κ are set such that Lemma 1 applies and avoid restating the corresponding parameter restriction for better readability.

Therefore, the statement follows immediately from the hardness of decision A-LWE$^f_{\kappa,\lambda,\sigma}$: A-LWE samples are indistinguishable from LWE$_{\kappa,\lambda,\sigma}$ samples due to [16]. Finally, A-LWE-samples are indistinguishable from uniform samples based on the hardness of worst-case lattice problems due to [39]. ■

The notion of aggregator obliviousness focuses on the aggregator's decryption capability: intuitively, it captures the requirement that the aggregator cannot learn anything from the participants' ciphertexts but the noisy sum, i.e. the aggregate. This also entails that retrieving the aggregate is not possible without the aggregator's decryption key. We present the formal statement of aggregator obliviousness security of our generic scheme in the following theorem.

Theorem 2 (Aggregator Obliviousness Security): Let the output of AHOM.Enc be indistinguishable from random and let $\sigma \geq 2\sqrt{\kappa}$. LaPS according to Definition 10 satisfies aggregator oblivious security assuming the hardness of worst case lattice problems.

Proof: See Appendix. ■

Privacy: Adding appropriately distributed noise to the users' raw values before encrypting them, constitutes a privacy mechanism and ensures the differential privacy of the aggregated output. The concrete differential privacy parameters that can be achieved are therefore entirely determined by the deployed privacy building block.

Below we formalize the concrete relation and state (ϵ, δ)-differential privacy of the aggregate output in terms of ϵ-differential privacy of the deployed privacy mechanism.

Theorem 3 (Privacy): Let $\mathcal{M}_\chi(D, f(\cdot), \epsilon) = f(D) + (Z_1, \ldots, Z_k)$ denote a mechanism for some function $f : \mathcal{D}^n \to \mathcal{R}^k$, where Z_i are i.i.d. random variables drawn according to some distribution χ.
If $\mathcal{M}_\chi$ achieves ϵ-DP and $f(D) = \sum_{i=1}^n d_i$ for $D = (d_1, \ldots, d_n)$ is a sum query, then the aggregate output generated by the LaPS according to Definition 10 is (ϵ, δ)-DP for $\beta = \min\{\frac{1}{\gamma n} \ln \frac{1}{\delta}, 1\}$, where n denotes the number of participants and γ is the fraction of honest participants.

[4]See Lemma 3 in [15] for a simplified version that is directly applied here.

Proof: The proof follows from [40, Lemma 1][5] when substituting their Geometric mechanism by $\mathcal{M}_\chi$. ■

Remark 2 (Trusted Setup): Our construction assumes a *trusted* setup as is customary for this type of PSA-protocol (see for instance [40], [8]), where all users and the aggregator are equipped with their respective secret keys as well as public parameters that are necessary to perform the protocol. In practice, this can be performed by executing the Setup routine (see Definition 10) with a trusted third party. Note that this is only needed once and no more interaction is required throughout the remainder of the protocol. In fact, it can be implemented in such a way that it does not need to be repeated for a new round of the protocol, for instance by dealing out multiple sets of secret key and other parameter information at once in advance.
Alternatively, as pointed out in [40] one may resort to standard secure MPC techniques in order to perform the setup in a distributed manner.

V. Example Instantiation

Starting with our generic PSA scheme LaPS from Definition 10, we can now pick an additively homomorphic scheme and a suitable privacy mechanism of our choice and construct an example instantiation of our scheme. More concretely, we choose a reduced version of the BGV scheme by Brakerski *et al.* [4] as the homomorphic scheme and the discrete version of the standard Laplace mechanism as the privacy mechanism.

In the following we first discuss these two components individually as stand-alone building blocks before giving a complete definition of the entire PSA scheme that results from embedding them into our generic scheme.

As mentioned previously, we use a Ring-A-LWE-based instantiation for efficiency reasons. Since we will implement our example scheme in Section VI, we already present our definitions here in the ring setting.

A. Additively homomorphic encryption: BGV

Originally introduced in [5], the BGV scheme was used by Damgård *et al.* [9] in the context of MPC. In this paper, we adapt their particular notation due to the fact that the original scheme is defined on binary bits and we are interested in a larger plaintext space. This also allows us to adopt their proof of correctness in a straightforward way.

Observe that we make several modifications to the formulation of the BGV-scheme in [9] for two major reasons. First of all, we do not require homomorphic multiplication: as we focus on sum aggregation we only need an additive homomorphism. Technically, this relaxes the requirement to a *somewhat homomorphic* encryption scheme. As a positive side effect, the scheme can be greatly simplified (eliminating parts such as key-switching) and efficiency is significantly improved, since multiplication is the most computationally intensive operation. Secondly, we do not generate the keys in a distributed manner as would be required for the MPC setting

[5]Note that in [40] the term *distributed* differential privacy is specifically coined for PSA schemes, since the privacy-preserving noise is generated in a distributed manner. For simplicity, we abuse notation and refer to "differential privacy".

in [9]. This affects the used distributions and the magnitudes of the resulting values. Furthermore, in order to optimize the parameter magnitudes we use the well-known reduction by Applebaum *et al.* [2] that lets us securely sample the secret in the R-LWE term from the error distribution as opposed to the uniform distribution.

We define the following subroutines as in [9]:

- $\mathcal{ZO}(0.5, n)$: Generate a vector of length n with elements chosen at random from $\{-1, 0, 1\}$ such that the probabilities for each coefficient are $p_{-1} = \frac{1}{4}$, $p_0 = \frac{1}{2}$, $p_1 = \frac{1}{4}$.
- $\mathcal{DG}(\sigma'^2, n)$: Generate a vector of length n with elements chosen according to the discrete Gaussian distribution with variance σ'^2.
- $\mathcal{RC}(0.5, \sigma'^2, n)$: Generate (v, e_0, e_1) where v is sampled from $\mathcal{ZO}(0.5, n)$ and e_0 and e_1 are sampled from $\mathcal{DG}(\sigma'^2, n)$.
- $\mathcal{U}(q, n)$: Generate a vector of length n with elements generated uniformly at random modulo q.

Observe that the BGV scheme uses *modulus-switching* in order to keep the error that arises due to arithmetic operations on ciphertexts in check. More concretely, given an input ciphertext (e.g. the result of adding two ciphertexts) defined with respect to some modulus q and the target modulus q', the SwitchModulus routine outputs a new ciphertext that is defined modulo q' but still encrypts the same underlying plaintext.

In contrast to the original definition, the SwitchModulus routine that we use in our definition below is equivalent to the function Scale in [19] as we do not explicitly perform an error estimation before reducing the input ciphertext (, which is part of the original SwitchModulus). Adopting Gentry *et al.*'s definition in [19]: $\mathsf{Scale}(x, q, q')$ takes an element $x \in R_q$ and returns an element $y \in R_{q'}$ such that in coefficient representation it holds that $y \equiv x \mod p$ and y is the closest element to $(q'/q) \cdot x$ that satisfies this mod-p condition. For a detailed description using evaluation representation refer to Appendix D in [19]. Finally, our adapted scheme is defined as follows.

Definition 11 (Adapted BGV [4], [9]): Let $R = \mathbb{Z}[X]/\Phi_m(X)$ and $R_q = (\mathbb{Z}/q\mathbb{Z})[X]/\Phi_m(X)$ for some cyclotomic polynomial $\Phi_m(X)$ and integer q, where $\phi(m)$ is the degree of R over $\mathbb{Z}$. Let σ' be the Gaussian standard deviation.
The plaintext space is R_p for some prime p and ciphertexts are tuples in $R_{q_1} \times R_{q_1}$, which get reduced (in the decryption process) to tuples in $R_{q_0} \times R_{q_0}$ for the two moduli q_0 and q_1. Set q_0, q_1 such that $q_0 = p_0$ and $q_1 = p_0 \cdot p_1$ for some primes p_0, p_1, where $q_0, q_1 > p$.

- $\mathsf{BGV.Gen}$: Generate $a \leftarrow \mathcal{U}(q_1, \phi(m))$. Draw $s, \epsilon \leftarrow \mathcal{DG}(\sigma'^2, \phi(m))$. Compute $b = a \cdot s + p \cdot \epsilon$ and output (a, b) as the public key and s as the secret key.
- $\mathsf{BGV.Enc}(pk, \mu \in R_p)$: Using modulus q_1, choose a "small" polynomial, i.e. with 0, ± 1 coefficients, and two polynomials with Gaussian coefficients $(v, e_0, e_1) \leftarrow \mathcal{RC}(0.5, \sigma'^2, \phi(m))$. Then set $c_0 = b \cdot v + p \cdot e_0 + \mu$, $c_1 = a \cdot v + p \cdot e_1$ and output ciphertext $c = (c_0, c_1) \in R_{q_1} \times R_{q_1}$.
- $\mathsf{BGV.Dec}(sk, c)$: For an input ciphertext c defined modulo q_1, invoke $\mathsf{SwitchModulus}(c, q_1, q_0)$, which produces a new ciphertext $c' = (c'_0, c'_1)$ defined modulo q_0 such that
$$\left((c'_0 - s \cdot c'_1) \mod q_0 \equiv (c_0 - s \cdot c_1) \mod q_1\right) \mod p.$$
Decryption of c' is performed by setting $m' = (c'_0 - s \cdot c'_1) \mod q_0$ and outputting $m' \mod p$.

Remark 3: For the remainder of this work we fix the following parameter choice following [9]: Choosing m as a power of 2 yields $\phi(m) = m/2$. Select $R = \mathbb{Z}[X]/(X^{m/2}+1)$ and $p = 1 \mod m$, i.e. $R_p \simeq \mathbb{F}_p^{m/2}$ and ring constant $c_m = 1$.

In the remainder of this section, we state semantic security and summarize the parameter requirements for correctness and concrete bit-security levels. The proofs follow from [9] with the necessary modifications for our BGV version. The interested reader is referred to the Appendix for the proof of correctness, which is more involved, and to the respective exposition in [9] for more details.

Lemma 2 (BGV: Semantic Security): The BGV scheme according to Definition 11 is semantically secure with pseudo-random ciphertexts assuming the hardness of decision Ring-LWE.

Proof: The proof follows from [9, Theorem 2] except that in our case the secret key is not generated in a distributed manner. Therefore we do not require the circular security assumption. Note that the ciphertexts are simply Ring-LWE samples and are thus indistinguishable from uniform samples as long as decision Ring-LWE is hard. ■

Parametrization for correctness and security: Adopting the analysis in [19], [9], the following inequality reflects the correctness requirement

$$\frac{N \cdot B_{clean}}{p_1} + B_{scale} < \frac{q_0}{2} = \frac{p_0}{2}, \tag{1}$$

where B_{clean} and B_{scale} are defined based on parameters $\phi(m)$, p and σ', see the Appendix for details.
Using the bit-security estimations from Lindner and Peikert [31] we get the following security requirement for ring degree $\phi(m)$, modulus q_1 and Gaussian parameter σ'

$$\phi(m) \geq \frac{(k + 110) \cdot \ln(q_1/\sigma')}{7.2}, \tag{2}$$

where k is the bit-security level.

B. Differential Privacy: Discrete Laplace Noise

Our second building block is a privacy mechanism that will serve to ensure differential privacy of the output aggregate. The (continuous) Laplace mechanism is a standard tool in differential privacy literature. Since our scheme works in the discrete world - as is common for cryptographic applications - we define a discrete Laplace Mechanism, which nevertheless provides the same differential privacy guarantees as we show below.

Definition 12 (The discrete Laplace Mechanism): Given any function $f : \mathcal{D}^n \rightarrow \mathcal{R}^k$, the discrete Laplace mechanism is defined as:

$$\mathcal{M}_{DLap}(D, f(\cdot), \epsilon) = f(D) + (Y_1, \ldots, Y_k)$$

where Y_i are i.i.d. random variables drawn from $DLap_\varsigma$ where $\varsigma = \Delta f/\epsilon$.

Theorem 4 (DLap-Mechanism: ϵ-DP): The discrete Laplace mechanism $\mathcal{M}_{DLap}$ preserves ϵ-DP.

Proof: The proof follows a standard structure that is common in differential privacy, see e.g. [14, Theorem 3.6]. In fact, it is analogous to the widely known continuous version of the Laplace mechanism with the exception of having a discrete function and distribution range. For completeness, we present the proof in the Appendix. ∎

C. Putting it together

Finally, we assemble the example instantiation of our LaPS scheme by plugging in the homomorphic scheme BGV described in Section V-A and the privacy-preserving mechanism from Section V-B into our generic Definition 10. As mentioned above, we define the instantiation in the ring setting such that the basis for our implementation in Section VI is clearly formulated. In consequence, we first explain the details of the underlying algebra before presenting the complete scheme and stating the correctness and security properties.

We use several moduli for different parts of the scheme: prime p is the plaintext modulus and q_1 denotes the overall ciphertext modulus. The decryption routine BGV.Dec within AggrDec internally produces reduced ciphertexts with modulus q_0. Set moduli $q_0, q_1 > p$ such that $q_0 = p_0$ and $q_1 = p_0 \cdot p_1$ for some primes p_0, p_1.

Based on these moduli we define the following rings: the plaintext space $R_p = (\mathbb{Z}/p\mathbb{Z})[X]/\Phi_{m'}(X)$, the internal key and ciphertext space $R_{\mathsf{int}} = (\mathbb{Z}/q_1\mathbb{Z})[X]/\Phi_{m'}(X)$ and the external key and ciphertext space $R_{\mathsf{ext}} = (\mathbb{Z}/q_1\mathbb{Z})[X]/\Phi_m(X)$ for some cyclotomic polynomials $\Phi_m(X)$ and $\Phi_{m'}(X)$. Internal ciphertexts are produced and processed by BGV routines and external ciphertexts are the actual outputs and inputs of the PSA scheme, i.e. the ciphertexts sent from the users to the aggregator.

Set m' to be a power of two and p s.t. $p \mod m' \equiv 1$, then the degree of R_p and R_{int} is $\phi(m') = \frac{m'}{2}$. Note that the only difference between R_{int} and R_{ext} is the dimension: namely choose[6] $\phi(m)$ s.t. $\phi(m) = 2 \cdot \phi(m') \cdot l$, where $l = \lceil \log q_1 \rceil$.

Note that we must transform from R_{ext} to $\mathbb{Z}_{q_1}$ in order to use Algorithm 1. For this purpose we define the following mappings:

- $\mathsf{z2R}_{q,m} : \mathbb{Z}_q \to R_q$: takes a scalar x over the q-ary field and produces a vector $y = (x, 0, \ldots, 0)$ of dimension $\phi(m)$, where y is coefficient representation for the output ring element.
- $\mathsf{R2z}_{q,m} : R_q \to \mathbb{Z}_q$: takes a ring element and outputs the first element of its coefficient representation.
- $\mathsf{R2Z}_{q,m} : R_q \to \mathbb{Z}_q^{\phi(m)}$: outputs a vector of size $\phi(m)$ by copying the entries of the coefficient representation of the input ring element.
- $\mathsf{Z2R}_{q,m} : \mathbb{Z}_q^{\phi(m)} \to R_q$: interprets the input vector as the coefficient representation of a polynomial in R_q and outputs the corresponding ring element.

In the following, observe that addition and multiplication of ring elements is performed component-wise. For better readability, we denote BGV-routine parameters and internal ciphertexts with bars above the variable names.

Definition 13 (LaPS using DLap and BGV): Let κ be a security parameter, $N \in \mathbb{N}$ the number of participants, γ the fraction of uncompromised users and let $\varsigma > 1$ and $\alpha \in (0, 1)$. Fix the rings R_p, R_{int} and R_{ext} with the corresponding parameters $p, p_0, p_1, q_0, q_1, m, m', l$ as described above. Let $\kappa = \phi(m)/l$.

- $(\{a, \mathbf{g}^T, pk\}, \{s_i\}, (sk_{A_1}, sk_{A_2})) \leftarrow \mathsf{Setup}(1^\kappa)$: Generate the public parameters a, $\mathbf{g}^T$ and pk as follows and distribute them to all parties.
  1) Draw a uniformly at random from R_{ext}.
  2) Set vector $\mathbf{g}^T = (1, 2, \ldots, 2^{l-1}) \in \mathbb{Z}_{q_1}^l$.
  3) Generate $((\bar{a}, \bar{b}), \bar{s}) \leftarrow \mathsf{BGV.Gen}$ and extract public key pk as $pk = (\bar{a}, \bar{b}) \in R_{\mathsf{int}} \times R_{\mathsf{int}}$.
  4) For all $i \in \{1, \ldots, N\}$ draw $s_i \leftarrow R_{\mathsf{ext}}$ and send it to user i as her secret key.
  5) The aggregator's secret decryption key is the tuple (sk_{A_1}, sk_{A_2}), where
     - $sk_{A_1} = -\sum_{i=1}^N s_i$ and
     - $sk_{A_2} = \bar{s} \in ((\bar{a}, \bar{b}), \bar{s}) \leftarrow \mathsf{BGV.Gen} \in R_{\mathsf{int}}$.
- $c_{i,a} \leftarrow \mathsf{NoisyEnc}_i(\{a, \mathbf{g}^T, pk\}, s_i, d_i)$: Each user i takes her data $d_{i,a} \in \mathcal{D}$ and adds some noise r_i to it, s.t. $x_i = d_i + r_i \mod p \in \mathbb{Z}_p$.
  1) r_i is sampled as follows:
     $$r_i = \begin{cases} 0 & \text{with probability } 1-\beta \\ Y & \text{with probability } \beta \end{cases},$$
     where $Y \leftarrow DLap_\varsigma$ and $\beta = \frac{1}{\gamma N} \log(\frac{1}{\delta})$.
  2) Set $\bar{x}_i = \mathsf{z2R}_{p,m'}(x_i) \in R_p$ and compute $\bar{c} = (\bar{c}_0, \bar{c}_1) \leftarrow \mathsf{BGV.Enc}(pk, \bar{x}_i)$.
  3) Set $\mathbf{v}_i = (\mathsf{R2Z}_{q_1,m'}(\bar{c}_0) || \mathsf{R2Z}_{q_1,m'}(\bar{c}_1)) \in \mathbb{Z}_{q_1}^{2\cdot\phi(m')}$.
  4) Invoke Algorithm 1 for each component of $\mathbf{v}_i$ in order to sample $\mathbf{e}_i \leftarrow D_{\Lambda_{\mathbf{v}_i}^\perp(\mathbf{G}),\sigma} \in \mathbb{Z}_{q_1}^{2\cdot\phi(m')\cdot l}$, i.e. $\mathbf{e}_i = (\mathsf{Sample}(\mathbf{g}^T, v_{i_1}, \sigma), \ldots, \mathsf{Sample}(\mathbf{g}^T, v_{i_{2\phi(m')}}, \sigma))$.
  5) Transform to the ring by $e_i = \mathsf{Z2R}_{q_1,m}(\mathbf{e}_i)$. Note that since $\phi(m) = 2 \cdot \phi(m') \cdot l$, $e_i \in R_{\mathsf{ext}}$.
  6) Output the ciphertext $c_{i,a} = a \cdot s_i + e_i \in R_{\mathsf{ext}}$.
- $\sum_{i=1}^N x_i \leftarrow \mathsf{AggrDec}(\{a, \mathbf{g}^T\}, (sk_{A_1}, sk_{A_2}), \{c_{1,a}, \ldots, c_{N,a}\})$: Receiving the users' ciphertexts $\{c_{i,a}\}$ the aggregator computes $c = \sum_{i=1}^N c_{i,a}$.
  1) Compute $e = \sum_{i=1}^N e_i = c + a \cdot sk_{A_1}$.
  2) Set $\mathbf{e} = \mathsf{R2Z}_{q_1,m}(e) \in \mathbb{Z}_{q_1}^{\phi(m)}$.
  3) Compute $\mathbf{v} = \mathbf{G} \cdot \mathbf{e} \mod q_1 \in \mathbb{Z}_{q_1}^{2\cdot\phi(m')}$, where $\mathbf{G} = \mathbf{I}_{\phi(m)/l} \otimes \mathbf{g}^T \in \mathbb{Z}_{q_1}^{\frac{\phi(m)}{l} \times \phi(m)}$.
     Again, note that $\phi(m)/l = 2 \cdot \phi(m')$ and that $\mathbf{v}$ is the sum of the individual $\mathbf{v}_i$'s from $\mathsf{NoisyEnc}_i$.
  4) Parse $\mathbf{v}$ as a tuple of vectors $\mathbf{v} = (\mathbf{v}', \mathbf{v}'') \in \mathbb{Z}_{q_1}^{\phi(m')} \times \mathbb{Z}_{q_1}^{\phi(m')}$.

[6] Note that with a small tweak to the described mappings Z2R and R2Z, $\phi(m)$ can also be made larger than $2 \cdot \phi(m') \cdot l$: simply add 0's to the coefficient representation to pad up to the desired length $\phi(m)$ in order to get a ring element in R_{ext} and cut the same number of 0's when transforming that ring element back to $\mathbb{Z}_{q_1}^{2\cdot\phi(m')\cdot l}$.

5) Decrypt: $\bar{x} = \mathsf{BGV.Dec}(sk_{A_2}, (\mathsf{Z2R}(\mathbf{v}'), \mathsf{Z2R}(\mathbf{v}''))) \in R_p$. Note that $\bar{x}$ is the sum of the individual $\bar{x}_i$'s from $\mathsf{NoisyEnc}_i$.
6) The aggregator retrieves the noisy sum of the users' values with $\sum_{i=1}^{N} x_i = \mathsf{R2z}_{p,m'}(\bar{x}) \in \mathbb{Z}_p$.

D. Correctness, Security and Privacy

The scheme inherits the security and privacy guarantees of our generic LaPS (see Section IV). Hence, we only need to show that the requirements of essentially Inequality (1) for correctness and Theorem 2 for aggregator obliviousness are met.

Correctness: The correctness of the PSA scheme stems from the correctness of the internal BGV scheme, since BGV.Dec determines the correct retrieval of the users' noisy sum. Therefore, in order to ensure correct decryption AggrDec, Inequality (1) must be satisfied with respect to R_{int}, i.e. fixing $\sigma' = 3.2$ as in [19], [9][7]

$$\frac{N \cdot B_{clean}}{p_1} + B_{scale} < \frac{q_0}{2} = \frac{p_0}{2}, \tag{3}$$

where $B_{clean} = \phi(m') \cdot (p-1) + 6.4p \cdot ((8+4\sqrt{2}) \cdot \phi(m') + 3 \cdot \sqrt{\phi(m')})$ and $B_{scale} = \frac{1}{\sqrt{3}} \cdot p \cdot (3 \cdot \sqrt{\phi(m')} + 3.2 \cdot \phi(m'))$.

Security: Below, we prove the security guarantees of our scheme.

Theorem 5 (Semantic Security): Let $\sigma' \geq \omega(1)$, $\epsilon = \mathsf{negl}(\kappa)$. The ciphertexts generated by NoisyEnc in the PSA scheme according to Definition 13 are semantically secure for $\sigma \geq \omega(\sqrt{\log(\kappa)}) \cdot (\kappa N / \log(\kappa N))^{\frac{1}{4}}$ assuming the hardness of worst case lattice problems.

Proof: Suppose, $\mathbf{v}_i$ that is generated in NoisyEnc in Step (3), is indistinguishable from random. Then, by Lemma 1 $D_{\Lambda^{\perp}_{\mathbf{v}_i}(\mathbf{G}),\sigma}$ correctly simulates the discrete Gaussian distribution $D_{\mathbb{Z}^{\phi(m)},\sigma}$ for $\sigma \geq \eta_\epsilon(\mathbf{G})$.
Note that $\mathbf{G} = \mathbf{I}_{\phi(m)/l} \otimes \mathbf{g}^T$. Adopting El Bansarkhani *et al.*'s [16] argumentation, $\mathbf{G}$ induces the lattice $\Lambda^{\perp}_{q_1}(\mathbf{G}) = \{\mathbf{x} \in \mathbb{Z}^{\phi(m)} | \mathbf{G}\mathbf{x} \equiv 0 \mod q_1\}$ with generator matrix $\mathbf{S} = \mathbf{I}_{\phi(m)/l} \otimes \mathbf{S}_l \in \mathbb{Z}_{q_1}^{\phi(m)/l \times l}$, where

$$\mathbf{S}_l = \begin{bmatrix} 2 & & & 0 \\ -1 & 2 & & \\ & \ddots & \ddots & \\ 0 & & -1 & 2 \end{bmatrix} \in \mathbb{Z}_{q_1}^{l \times l}.$$

Using Lemma 3 in [15] (derived from Lemma 2.6 in [20]), which states an upper bound on the smoothing parameter $\eta_\epsilon(\Lambda)$ of a given lattice Λ and its basis, the smoothing parameter $\eta_\epsilon(\Lambda^{\perp}_{q_1})$ is bounded from above by $||\mathbf{S}|| \cdot \sqrt{\ln(2\frac{\phi(m)}{l}(1+\frac{1}{\epsilon}))/\pi} \leq 2 \cdot \sqrt{\phi(m)/l}$. By definition $\kappa = \phi(m)/l$ and hence $\eta_\epsilon(\Lambda^{\perp}_{q_1}) \leq 2 \cdot \sqrt{\kappa}$.

Consequently, Lemma 1 applies and the ciphertexts $c_{i,a}$ represent plain Ring-A-LWE samples. Observe that the hardness of decision A-LWE carries over to the ring setting in the straightforward way[8] as shown in [15]. Therefore, the statement follows immediately from the hardness of decision Ring-A-LWE: Ring-A-LWE samples are indistinguishable from Ring-LWE samples due to [16]. Finally, decision Ring-LWE is hard based on the hardness of worst-case lattice problems [37]. The latter holds as long as $\sigma \geq \omega(\sqrt{\log(\kappa)}) \cdot (\kappa N / \log(\kappa N))^{\frac{1}{4}}$, which is given by assumption. Observe that this parameter constraint arises from Corollary 7.3 in [37], which states the hardness assumption for decision Ring-LWE with spherical error[9].

Finally, note that $\mathbf{v}_i$ in Step (3) is the ring-transform of internal ciphertext $\bar{c}$ from Step (2). Semantic security and pseudo-randomness of internal ciphertexts follows from Lemma 2 assuming the hardness of Ring-LWE. Setting $\sigma' = \alpha' q_1$ and Since $\sigma' \geq \omega(1)$, hardness of decision Ring-LWE is satisfied according to Theorem 6.2 in [37] and $\mathbf{v}_i$ is indeed indistinguishable from random and subsequently the claim follows. ■

Aggregator obliviousness is inherited from our generic LaPS scheme (see Section IV) assuming pseudo-randomness of the internal ciphertexts. Since the latter is part of the semantic security of the overall scheme and we indeed prove this property for our example instantiation in Theorem 5, aggregator obliviousness security of our example instantiation follows immediately. We state the corresponding theorem below.

Theorem 6 (Aggregator Obliviousness Security): Let parameters be as in Theorem 5. Then, the PSA scheme according to Definition 13 satisfies aggregator oblivious security assuming the hardness of worst case lattice problems.

Proof: This follows directly from aggregator obliviousness of LaPS due to Theorem 2 when applied to the ring setting and semantic security from Theorem 5. ■

Privacy & Accuracy of Aggregate Output: Suppose that γ is the fraction of honest participants in the aggregation. Then, assuming that all values in the data domain $\mathcal{D}$ are inside an interval of width Δ in $\mathbb{Z}_p$, γN users each add noise with magnitude $\Theta(\frac{\Delta}{\epsilon})$. In the final aggregate output this accumulates to a total noise of roughly $O(\frac{\Delta}{\epsilon}\sqrt{N})$ magnitude according to Shi *et al.*'s [40] randomization procedure. Formally, we give the following privacy and accuracy guarantees.

Theorem 7: Let $\epsilon > 0$, $0 < \delta < 1$, $\Delta \geq \frac{\epsilon}{3}$, $\gamma \geq \frac{1}{N}\ln(\frac{1}{\delta})$ and $\varsigma = \Delta/\epsilon$. The output of AggrDec as in Definition 13 is (ϵ, δ)-differentially private. Moreover, the aggregate achieves $(\frac{4\Delta}{\epsilon}\sqrt{\frac{1}{\gamma}\ln(\frac{1}{\delta})\ln(\frac{2}{\eta})}, \eta)$-accuracy for all η s.t. $\ln(\frac{2}{\eta}) \leq \frac{1}{\gamma}\ln(\frac{1}{\delta})$.

Proof: With the chosen parameters it is straightforward to recognize the randomization procedure in NoisyEnc according to Definition 13 as a randomized Discrete Laplace mechanism $\mathcal{M}_{DLap}$ as in Definition 12, where the function f is the sum function. (ϵ, δ)-differential privacy thus follows immediately from ϵ-DP of $\mathcal{M}_{DLap}$ due to Theorem 4 and differential privacy of LaPS due to Theorem 3. Lastly, (α, β)-accuracy follows from utility of Shi *et al.*'s [40] randomization procedure. ■

[7] Gentry *et al.* [19] originally choose this value according to the parameter analysis from Micciancio and Regev [35].

[8] See Section 4.5 in [15] for an example of a BGV-based encryption scheme that is reduced to the ring variant of A-LWE.

[9] The main theorem in [37] is not applicable here as the error terms resulting from the sampling routine Sample are spherical.

VI. Practical Performance

In this section we provide example parameter sets satisfying correctness and security requirements and present experimental results based on our implementation of the PSA scheme according to Definition 13. Finally, we carefully compare the practical performance of our scheme to the most closely related prior schemes from [40] and [42] in order to put our results into context.

A. Example parameters

Note that the parameter analysis for concrete bit-security levels in the context of the BGV scheme also applies here since security is based on Ring-LWE in both cases. Hence, taking Inequality (2) and applying it to the parameters corresponding to R_{ext}, i.e. for Gaussian parameter σ, we get

$$\phi(m) \geq \frac{(k+110)\cdot \ln(q_1/\sigma)}{7.2}, \tag{4}$$

where k is the bit-security level.
Combining the correctness requirement from Inequality (3), the hardness requirement from Theorem 5 and the bit-security estimate from Inequality (4) yields the set of constraints for valid parameters. Below we provide an overview of possible parameter sets for varying security level k, plaintext modulus p and number of users N.

Since above constraints yield a few circular dependencies on the parameters, we calculated the remaining parameters by fixing $\sigma = 0.1q_1$ and first choosing p_1 before picking[10] $\phi(m')$. With these parameters at hand we plug in into Inequality (3) and get p_0, which gives a concrete value for q_1, since $q_1 = p_0 \cdot p_1$. Looking at Damgård *et al.*'s [9] instantiation of

N	$\log(p_0 = q_0)$	$\log(q_1 = p_0 \cdot p_1)$
100	31	36
1000	34	39
10000	38	43

Fig. 3: Parameters for $p \approx 2^{16}$ and $k = 80$, $\phi(m') = 32$, $\phi(m) \approx 2^{11}$, $p_1 \approx 2^5$

N	$\log(p_0 = q_0)$	$\log(q_1 = p_0 \cdot p_1)$
100	48	63
1000	49	64
10000	52	67

Fig. 4: Parameters for $p \approx 2^{32}$ and $k = 128$, $\phi(m') = 8192$, $\phi(m) \approx 2^{20}$, $p_1 \approx 2^{15}$

N	$\log(p_0 = q_0)$	$\log(q_1 = p_0 \cdot p_1)$
10000	146	196
10^{15}	151	201
10^{21}	171	221

Fig. 5: Parameters for $p \approx 2^{128}$ and $k = 80$, $\phi(m') = 32768$, $\phi(m) \approx 2^{24}$, $p_1 \approx 2^{50}$

the BGV scheme for the same parameters k, p and $\phi(m')$, our instantiation (see Figure 4) allows for much smaller moduli, i.e. our q_1 has magnitude 2^{63} for 100 users compared to 2^{252} in [9, Appendix G.4]. This improvement draws from the highly relaxed correctness requirement, i.e. Inequality (1). Since we do not require correct evaluation of homomorphic multiplication our constraint is much less restrictive.

Note that for large enough moduli p and q_1 the number of users can grow to a large extent without significantly affecting the other parameters as shown in Figure 5. It is in that sense therefore highly scalable in the number of participants.

B. Implementation

We implemented our PSA scheme according to Definition 13 on a MacBook running macOS Sierra with a single 2.5 GHz Intel Core i7 and 16GB memory using part of the HElib C++ library[11].

	p	NoisyEnc (ms)	AggrDec (ms)
(i)	$5 \approx 2^2$	3.57646	1.86864
(i)	$37 \approx 2^5$	3.61646	1.882
(i)	$65537 \approx 2^{16}$	3.72438	1.96416
(ii)	$65537 \approx 2^{16}$	77.3304	67.6243

Fig. 6: Results for $N = 1000$,
(i) $k = 80$, $\phi(m') = 32$, $\phi(m) = 2048$,
(ii) $k = 128$, $\phi(m') = 512$, $\phi(m) \approx 2^{15}, q_1 \approx 2^{44}$

We use a parameter set satisfying both correctness and security requirements for $k = 80$-bit security, i.e. with magnitudes according to Figure 3, and a selection of plaintext spaces $p \leq 2^{16}$ for setting (i). Note that this parameter selection can be used for example in order to compute the average age of 1000 people assuming that each person is at most 65 years old[12]. We measured the average encryption and decryption time over 1000 runs each for $N = 1000$ participants in milliseconds. For completeness, we also ran experiments for $k = 128$-bit security and plaintext space $p \approx 2^{16}$ (setting (ii)). The results along with the concrete parameters are shown in Figure 6. We chose target DP-parameters $\epsilon = 1$, $\delta = 0.1$, which results in a minimum fraction of honest participants $\gamma \geq 0.0023$, i.e. at least 3 out of 1000 participants have to remain uncompromised. Finally, due to Theorem 7 and when choosing $\eta = 2/\exp(10)$ we achieve $(400 \cdot (p-1), 2/\exp(10))$-accuracy, where the accumulated sum does not exceed $1000 \cdot (p-1)$.

To the best of our knowledge none of the previously proposed PSA schemes that are comparable to ours[13], including the scheme introduced by Shi *et al.* [40] and the LWE-based scheme from Valovich [42], were implemented. Nevertheless both works provide rough guidelines for evaluating accuracy of the final output and performance of the scheme in practice, which we use as a baseline for comparison.

[10] Note that the first two constraints depend on q_1 but are easily satisfied as long as $q_1 \gg \phi(m')$.

[11] https://github.com/shaih/HElib

[12] According to [7] the majority of Americans, i.e. 85.9 %, is under 65 years old.

[13] Runtime measurements are given in [29] based on implementation results. However, their scheme accounts for dynamic user leaves and joins, which our scheme does not currently address. Therefore we do not consider their results to be comparable.

Valovich and Alda [43] compare the accuracy of different noise distributions when embedded into a privacy-preserving mechanism: they conclude that the Geometric mechanism, i.e. our $\mathcal{M}_{DLap}$ according to Definition 12, and the Skellam mechanism, which is used in Valovich's [42] scheme, are comparable in terms of accuracy (while using the Binomial distribution is significantly less accurate). As mentioned before we cannot compare our scheme to theirs in terms of runtime since no experimental results are available to this extent.

Shi *et al.* [40] estimate the time it takes to encrypt with roughly 6ms when using a classic Diffie-Hellman group modular a 1024-bit prime or roughly 0.6 ms using high-speed elliptic curves. While we clearly outperform the former (compare at parameter setting (i)), our scheme would be slower than an elliptic-curve based implementation of Shi *et al.*'s PSA scheme by a factor of roughly 6. On the other hand, the decryption operation in their scheme takes about 0.3 s for 1000 participants using a brute-force approach. Hence, our decryption is more than 150 times faster. Additionally, note that their result is based on the assumption that each participant encrypts a single bit, whereas our plaintext space is much larger, i.e. we allow for encryption of up to 16 bits. Lastly, while we don't have a direct comparison for the 128-bit security level, our experimental results indicate practical runtimes for this setting as well.

VII. Extensions

Prior work has produced some extensions to the standard sum aggregation case. Concretely, [40] describes how to adapt their scheme in order to allow for the evaluation of user *data distribution*, enable *public access* to the aggregate output, i.e. aggregation without an explicit aggregating party, or support hierarchical *access control* to the aggregates of subsets of the user data.

Chan *et al.* [8] develop on Shi *et al.*'s [40] scheme and address the problem of *user failures*, i.e. ensuring correct decryption even when some users do not participate in the aggregation. They achieve this by essentially grouping the users into sub-groups and structuring these into a binary tree at some cost of the accuracy of the output and communication overhead. We expect that the same techniques would also apply to our scheme in a straightforward way.

In principle, our scheme can also be extended to computing the *product aggregate*. This would require a multiplicatively homomorphic encryption scheme as the respective building block[14]. In fact, if a fully homomorphic encryption scheme is used, both addition and multiplication and therefore *richer statistics* could be supported. Since we circumvent the problem of using homomorphic operations under multiple keys, this would address the corresponding open problem from Shi *et al.*'s [40]. However, the extent of the statistical capabilities remains constrained by the limits of the used homomorphic scheme (and requires some additional adjustments in the formulation of the aggregator's decryption key).

[14]Additionally, the aggregator's decryption key needs to be defined slightly differently: it has to be the multiplicative inverse of the product of the users' secret keys.

VIII. Conclusion

Private stream aggregation captures the problem of performing aggregation operations over data from different users in a privacy-friendly manner without a trusted aggregator. In this work we presented LaPS that in contrast to most prior work provides privacy and security guarantees even against quantum adversaries. We formulated our scheme in a generic way that allows for a flexible choice of homomorphic encryption and privacy-preserving mechanism depending on the particular application's needs while maintaining strong security and privacy guarantees. In fact, our formal security analysis proves that we are additionally able to abolish assumptions from previously defined security models, such as the encrypt-once model required by [40]. Furthermore, we presented a concrete instantiation using the BGV somewhat homomorphic encryption scheme and the discrete Laplace mechanism. Our implementation results show that our scheme is compact and efficient: it significantly improves over prior work in terms of run time.

Acknowledgment

The authors would like to thank the reviewers and the shepherd for valuable comments and Noah Stephens-Davidowitz for helpful discussion.

References

[1] M. Antikainen, T. Aura, and M. Särelä, "Denial-of-Service Attacks in Bloom-Filter-Based Forwarding," *IEEE/ACM Trans. Netw.*, vol. 22, no. 5, pp. 1463–1476, 2014. [Online]. Available: http://dx.doi.org/10.1109/TNET.2013.2281614

[2] B. Applebaum, D. Cash, C. Peikert, and A. Sahai, "Fast cryptographic primitives and circular-secure encryption based on hard learning problems," in *CRYPTO '09 Proc. 29th Annu. Int. Cryptol. Conf. Adv. Cryptol.*, 2009, pp. 595–618.

[3] G. Asharov, A. Jain, and D. Wichs, "Multiparty computation with low communication, computation and interaction via threshold FHE," in *Adv. Cryptol. - EUROCRYPT 2012*, vol. 7237 LNCS, no. 1017660, 2012, pp. 483–501.

[4] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "Fully Homomorphic Encryption without Bootstrapping," in *Proc. 3rd Innov. Theor. Comput. Sci. Conf.*, 2012, pp. 309–325.

[5] Z. Brakerski and V. Vaikuntanathan, "Efficient Fully Homomorphic Encryption from (Standard) LWE," in *Proc. 2011 IEEE 52nd Annu. Symp. Found. Comput. Sci.*, 2011, pp. 97–106.

[6] ——, "Fully homomorphic encryption from ring-LWE and security for key dependent messages," *CRYPTO 2011*, pp. 505–524, 2011.

[7] U. C. Bureau, "American FactFinder," https://factfinder.census.gov/faces/tableservices/jsf/pages/productview.xhtml?pid=ACS_14_5YR_DP05&src=pt, Tech. Rep., 2015.

[8] T.-H. H. Chan, E. Shi, and D. Song, "Privacy-Preserving Stream Aggregation with Fault Tolerance," in *Int. Conf. Financ. Cryptogr. Data Secur.*, no. 1, 2012, pp. 200–214.

[9] I. Damgård, M. Keller, E. Larraia, V. Pastro, P. Scholl, and N. P. Smart, "Practical Covertly Secure MPC for Dishonest Majority or : Breaking the SPDZ Limits," in *Eur. Symp. Res. Comput. Secur.*, 2013, pp. 1–18.

[10] G. Danezis, C. Fournet, M. Kohlweiss, and S. Zanella-Béguelin, "Smart meter aggregation via secret-sharing," *Proc. first ACM Work. Smart energy grid Secur. - SEGS '13*, pp. 75–80.

[11] J. Ding, X. Xie, and X. Lin, "A Simple Provably Secure Key Exchange Scheme Based on the Learning with Errors Problem," *IACR Cryptol. ePrint Arch.*

[12] N. Döttling and J. Müller-Quade, "Lossy codes and a new variant of the learning-with-errors problem," in *Adv. Cryptol. - EUROCRYPT 2013*, 2013, pp. 18–34.

[13] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, "Local privacy and statistical minimax rates," in *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, Oct 2013, pp. 429–438.

[14] C. Dwork and A. Roth, "The Algorithmic Foundations of Differential Privacy," in *Found. Trends Theor. Comput. Sci.*, 2014, vol. 9, pp. 211–407.

[15] R. El Bansarkhani, Ö. Dagdelen, and J. Buchmann, "Augmented Learning with Errors: The Untapped Potential of the Error Term (Full Version)," *Eprint*, 2014.

[16] ——, "Augmented Learning with Errors: The Untapped Potential of the Error Term," in *19th Int. Conf. FC 2015*, vol. 8975, 2015, pp. 333–352.

[17] U. Erlingsson, V. Pihur, and A. Korolova, "Rappor: Randomized aggregatable privacy-preserving ordinal response," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '14. New York, NY, USA: ACM, 2014, pp. 1054–1067. [Online]. Available: http://doi.acm.org/10.1145/2660267.2660348

[18] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, "Privacy in Pharmacogenetics: An End-to-End Case Study of Personalized Warfarin Dosing," in *Proc. 23rd USENIX Secur. Symp.*, pp. 17–32.

[19] C. Gentry, S. Halevi, and N. P. Smart, "Homomorphic Evaluation of the AES Circuit (Updated Implementation)," in *Adv. Cryptol. - CRYPTO 2012*, 2012, pp. 850–867.

[20] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for Hard Lattices and New Cryptographic Constructions," in *Proc. 40th Annu. ACM Symp. Theory Comput.*, 2008, pp. 197–206.

[21] C. Gentry, A. Sahai, and B. Waters, "Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based," in *Adv. Cryptol. - CRYPTO 2013*, 2013, pp. 75 – 92.

[22] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game," in *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, ser. STOC '87, 1987, pp. 218–229.

[23] O. Goldreich, "Secure Multi-Party Computation," http://www.wisdom.weizmann.ac.il/{~}odedg/pp.html, pp. 1–110, 1998.

[24] S. Inusah and T. J. Kozubowski, "A discrete analogue of the Laplace distribution," *J. Stat. Plan. Inference*, vol. 136, pp. 1090–1102, 2006.

[25] M. Jawurek, F. Kerschbaum, and G. Danezis, "SoK : Privacy Technologies for Smart Grids - A Survey of Options." p. 16.

[26] T. Jung, X. Mao, X. Y. Li, S. J. Tang, W. Gong, and L. Zhang, "Privacy-preserving data aggregation without secure channel: Multivariate polynomial evaluation," in *Proc. - IEEE INFOCOM*, no. 61170216, 2013, pp. 2634–2642.

[27] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith, "What Can We Learn Privately?" in *Proc. - Annu. IEEE Symp. Found. Comput. Sci. FOCS*, 2008, pp. 531–540.

[28] J. Katz and V. Vaikuntanathan, *Smooth Projective Hashing and Password-Based Authenticated Key Exchange from Lattices*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 636–652. [Online]. Available: https://doi.org/10.1007/978-3-642-10366-7_37

[29] Q. Li and G. Cao, "Efficient Privacy-Preserving Stream Aggregation in Mobile Sensing with Low Aggregation Error," in *Int. Symp. Priv. Enhancing Technol. Symp. - PETS 2013*, 2013, pp. 60–81.

[30] Q. Li, G. Cao, and T. F. La Porta, "Efficient and privacy-aware data aggregation in mobile sensing," *IEEE Trans. Dependable Secur. Comput.*, vol. 11, no. 2, pp. 115–129, 2014.

[31] R. Lindner and C. Peikert, "Better key sizes (and Attacks) for LWE-based encryption," in *Top. Cryptol. - CT-RSA 2011*, 2011, pp. 319–339.

[32] V. Lyubashevsky, C. Peikert, and O. Regev, "On Ideal Lattices and Learning with Errors over Rings," in *Eurocrypt 2010*, 2010.

[33] ——, "A toolkit for ring-LWE cryptography," in *EUROCRYPT 2013 Adv. Cryptol.*, vol. 7881 LNCS, 2013, pp. 35–54.

[34] D. Micciancio and C. Peikert, "Trapdoors for lattices: Simpler, tighter, faster, smaller," in *Adv. Cryptol. - EUROCRYPT 2012*, 2012, pp. 700–718.

[35] D. Micciancio and O. Regev, "Lattice-based Cryptography," in *Post-quantum Cryptogr.* Springer, 2009, pp. 147–192.

[36] F. Niedermeyer, S. Steinmetzer, M. Kroll, and R. Schnell, "Cryptanalysis of Basic Bloom Filters used for Privacy Preserving Record Linkage," *J. Priv. Confidentiality*, vol. 6, no. 2, pp. 59–79, 2014.

[37] C. Peikert, O. Regev, and N. Stephens-Davidowitz, "Pseudorandomness of Ring-LWE for Any Ring and Modulus," in *Proc. 49th Annu. ACM SIGACT Symp. Theory Comput. STOC 2017*, 2017, pp. 461—-473. [Online]. Available: https://eprint.iacr.org/2017/258.pdf

[38] C. Peikert and B. Waters, "Lossy Trapdoor Functions and Their Applications," in *STOC '08 Proc. fortieth Annu. ACM Symp. Theory Comput.*, 2008, pp. 187–196.

[39] O. Regev, "On Lattices, Learning with Errors, Random Linear codes, and Cryptography," *J. ACM*, vol. 56, no. 6, pp. 1–40, 2009.

[40] E. Shi, T.-H. H. Chan, E. Rieffel, R. Chow, and D. Song, "Privacy-Preserving Aggregation of Time-Series Data," in *Proceedings of the Network and Distributed System Security Symposium, NDSS 2011*, 2011.

[41] J. Ullman and S. Vadhan, *PCPs and the Hardness of Generating Private Synthetic Data*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 400–416. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-19571-6_24

[42] F. Valovich, "On the hardness of the Learning with Errors problem with a discrete reproducible error distribution," *arXiv*, 2016.

[43] F. Valovich and F. Alda, "Private Stream Aggregation Revisited," *arXiv*, 2015.

[44] S. L. Warner, "Randomized response: A survey technique for eliminating evasive answer bias," *Journal of the American Statistical Association*, vol. 60, no. 309, pp. 63–69, 1965, pMID: 12261830. [Online]. Available: http://amstat.tandfonline.com/doi/abs/10.1080/01621459.1965.10480775

[45] A. C.-C. Yao, "Protocols for secure computations," in *23rd Annu. Symp. Found. Comput. Sci. (sfcs 1982)*, 1982, pp. 1–5.

[46] ——, "How to generate and exchange secrets," in *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, ser. SFCS '86, 1986, pp. 162–167.

APPENDIX

Proof of Theorem 4: Let $D_0 \in \mathcal{D}^n$ and $D_1 \in \mathcal{D}^n$ be adjacent databases, and let $f(\cdot)$ be some function $f : \mathcal{D}^n \to \mathcal{R}^k$ and $\Delta f = \max_{D_0, D_1 \text{ adjacent}} \|f(D_0) - f(D_1)\|_1$. Comparison at some arbitrary point $z \in \mathcal{R}^k$ yields

$$\begin{aligned}
\frac{\Pr[\mathcal{M}_{DLap}(D_0, f, \epsilon) = z]}{\Pr[\mathcal{M}_{DLap}(D_1, f, \epsilon) = z]} &= \prod_{i=1}^{k} \left(\frac{\exp(-\frac{\epsilon|z_i - f(D_0)_i|}{\Delta f})}{\exp(-\frac{\epsilon|z_i - f(D_1)_i|}{\Delta f})} \right) \\
&= \prod_{i=1}^{k} \exp\left(\frac{\epsilon(|z_i - f(D_1)_i| - |z_i - f(D_0)_i|)}{\Delta f} \right) \\
&\leq \prod_{i=1}^{k} \exp\left(\frac{\epsilon|f(D_0)_i - f(D_1)_i|}{\Delta f} \right) \\
&= \exp\left(\frac{\epsilon\|f(D_0) - f(D_1)\|_1}{\Delta f} \right) \leq \exp(\epsilon)
\end{aligned}$$

Similarly, $\frac{\Pr[\mathcal{M}_{DLap}(D_1, f, \epsilon) = z]}{\Pr[\mathcal{M}_{DLap}(D_0, f, \epsilon) = z]} \geq \exp(-\epsilon)$ by symmetry. ■

Proof of Theorem 2: Note that this property (together with Theorem 1) targets the security of the PSA scheme as opposed to its privacy. It is independent of the used randomization procedure that adds noise to the users' values. We therefore assume that a potential adversary can choose the noise r_i as part of the **Challenge** phase in the respective security game as specified in Definition 6. Concretely, we adopt the notation $\mathsf{NoisyEnc}_i(pk, \mathbf{g}^T, \mathbf{s}_i, \mathbf{A}, d_i, r_i)$ for directly setting $x_i = d_i + r_i \mod q$ and encrypting x_i. This is in line with previous work, such as Shi *et al.*'s PSA scheme, which

is proven to be aggregator oblivious independent of the used randomization procedure [40].

Using Theorem 1 above it suffices to show: If there exists a PPT adversary $\mathcal{A}$ that wins the aggregator obliviousness security game, then there exists a PPT adversary $\mathcal{B}$ that can solve the decision $\mathsf{A\text{-}LWE}^{f}_{\kappa,\lambda,\sigma}$, i.e. distinguish an $\mathsf{A\text{-}LWE}_{\kappa,\lambda,\sigma}$ sample from a uniformly random sample over $\mathbb{Z}_q^{\kappa\times\lambda} \times \mathbb{Z}_q^{\lambda}$.

We define the following intermediate game Game similar to [40] that is indistinguishable from the aggregator obliviousness security game according to Definition 6: First, we treat any **Encrypt** query as a **Compromise** query from the adversary. Clearly, this turns the adversary actually more powerful. Secondly, we change the **Challenge** phase to its real-or-random version, i.e. instead of having the adversary specify two sets of plaintext-randomness pairs $\{(d_i, r_i)\}$ and $\{(d'_i, r'_i)\}$ and have her distinguish between encryptions of either one, we let the adversary pick one set $\{(d_i, r_i)\}$ and have her distinguish between a set of valid encryptions and a set of random values in $\mathbb{Z}_q^{\lambda}$. It is straightforward that any adversary with more than negligible advantage in winning Game will also win the aggregator obliviousness security game with more than negligible advantage. Therefore, it suffices to show that with a PPT adversary $\mathcal{A}$ with more than negligible advantage in winning Game we can construct an algorithm $\mathcal{B}$ that solves decision $\mathsf{A\text{-}LWE}^{f}_{\kappa,\lambda,\sigma}$ with more than negligible advantage.

Remark 4: Note that in the case where the adversary compromises all but one participant, she inevitably learns the secret key of that participant and can therefore distinguish between valid encryptions and random values. In this case the definition of aggregator obliviousness requires that she does not learn any additional information about that participant. As stated in Definition 6 this requirement translates into a privacy rather than a security guarantee. Therefore, we address it in the corresponding Theorem 3.

Suppose $\mathcal{B}$ receives the parameters κ, λ, σ and function f and plays the standard real-or-random game with challenger $\mathcal{C}$ who tests $\mathcal{B}$'s ability of solving the decision $\mathsf{A\text{-}LWE}$ problem. Hence $\mathcal{C}$ possesses an $\mathsf{A\text{-}LWE}$ distribution $L^{\mathsf{A\text{-}LWE}}_{\kappa,\lambda,\sigma}$ and can generate $\mathsf{A\text{-}LWE}$ samples $(\mathbf{A}, \mathbf{b}^T = \mathbf{s}^{*T}\mathbf{A} + \mathbf{e}^T)$ for some $m \in \mathbb{Z}_q$, where $\mathbf{s}^* \in \mathbb{Z}_q^{\kappa}$ is the secret, $\mathbf{A}$ is a public matrix in $\mathbb{Z}_q^{\kappa\times\lambda}$ and the error term $\mathbf{e} \in \mathbb{Z}_q^{\lambda}$ embeds the message m. In the game $\mathcal{B}$ is allowed to make **Sample** queries, where she provides an $m \in \mathbb{Z}_q$ to $\mathcal{C}$ who generates an $\mathsf{A\text{-}LWE}$ sample from L accordingly and returns it to $\mathcal{B}$. In the **Distinguish** phase $\mathcal{B}$ picks a new message $m^* \in \mathbb{Z}_q$ and sends it to $\mathcal{C}$. $\mathcal{C}$ then flips a random coin b: if $b = 0$, generate a valid $\mathsf{A\text{-}LWE}$ sample embedding m^* from L, otherwise draw $\mathbf{b}$ uniformly at random from $\mathbb{Z}_q^{\lambda}$ and send the tuple $(\mathbf{A}, \mathbf{b}^T)$ to $\mathcal{B}$. Finally, $\mathcal{B}$ outputs her guess whether b was 0 or 1. She wins the game and hence solves decision $\mathsf{A\text{-}LWE}^{f}_{\kappa,\lambda,\sigma}$ if her guess of b is correct. Note that f is defined as $\mathsf{AHOM.Enc}$, where $\mathsf{AHOM} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$.

Setup. $\mathcal{B}$ takes $\mathbf{A}$, $\mathbf{g}^T = (1, 2, \ldots, 2^{\lambda-1})$ and $(sk, pk) \leftarrow \mathsf{AHOM.Gen}$ that she has received from prior interaction with $\mathcal{C}$ and sends $\mathbf{A}, \mathbf{g}^T$ and pk to $\mathcal{A}$ as the public parameters. Then, $\mathcal{B}$ randomly chooses two distinct indices j and k from $\{0, \ldots, N\}$. The chance of picking the right ones is $\frac{1}{N^2}$, otherwise $\mathcal{B}$ aborts during the game. She sets

$$\mathbf{s}_k := ((\mathbf{b}^T - \mathbf{e}^T)\mathbf{A}^{-1})^T.$$

Hence, $\mathbf{s}_k$ is the secret $\mathbf{s}^*$. Note that $\mathbf{s}_k$ is *unknown* to $\mathcal{B}$. $\mathcal{B}$ samples the users' secret keys $\{\mathbf{s}_i \xleftarrow{\$} \mathbb{Z}_q^{\kappa}\}_{i\neq k}^{i\neq j}$ and sets

$$\mathbf{s}_j := -\sum_{i\neq j} \mathbf{s}_i = -(\sum_{\substack{i\neq j,\\ i\neq k}} \mathbf{s}_i + \mathbf{s}_k).$$

This comes from the inherent requirement of the PSA protocol $\sum_{i=0}^{N} \mathbf{s}_i = \mathbf{0}$. Note that $\mathbf{s}_j$ is also *unknown* to $\mathcal{B}$.

Lastly, $\mathcal{B}$ picks $sk_{A_1} \xleftarrow{\$} \{\mathbf{s}_i\}_{i\neq k}^{i\neq j}$ and sets $sk_{A_2} := sk$. Note that for the first aggregator key it indeed does not matter which $\mathbf{s}_i$ is chosen, since we ensure that $\sum_{i=0}^{N} \mathbf{s}_i = \mathbf{0}$ by choosing $\mathbf{s}_j$ according to Equation (A).

Compromise. On request i from $\mathcal{A}$ and if $i \neq j$, $i \neq k$ $\mathcal{B}$ sends the corresponding $\mathbf{s}_i$ to $\mathcal{A}$. If additionally $i = 0$ $\mathcal{B}$ sends (sk_{A_1}, sk_{A_2}). Otherwise $\mathcal{B}$ aborts. Let K be the set of all compromised users $K = \{i\}$.

Challenge. $\mathcal{A}$ picks a set of uncompromised users $U \subseteq \{0, \ldots, N\}\backslash K$ and plaintext-randomness pairs $\{(d_i, r_i)\}^{i\in U}$ and transmits these to $\mathcal{B}$. Note that by construction $\{j, k\} \subseteq U$. $\mathcal{B}$ computes

$$\{\mathbf{c}_i = \mathsf{NoisyEnc}(pk, \mathbf{g}^T, \mathbf{s}_i, \mathbf{A}, d_i, r_i)\}^{i\in U\backslash\{j,k\}}.$$

Now $\mathcal{B}$ enters the **Distinguish**-phase and sends $m = d_k + r_k \mod q$ to $\mathcal{C}$ who returns the tuple $(\mathbf{A}, \mathbf{b}^T)$. $\mathcal{B}$ sets $\mathbf{c}_k := \mathbf{b}^T$. Note that $\sum_{i\in U} \mathbf{c}_i + \sum_{i\notin U} \mathbf{s}_i^T \cdot \mathbf{A} = \sum_{i\in U} \mathbf{e}_i$ and that

$$\mathsf{AHOM.Dec}(\mathbf{G} \cdot \sum_{i\in U} \mathbf{e}_i \mod q) \stackrel{!}{=} \sum_{i\in U} d_i + r_i.$$

Therefore, $\mathcal{B}$ first computes a valid encryption of $\sum_{i\in U} d_i + r_i$, i.e.

$$\mathbf{v} = \mathsf{AHOM.Enc}(pk, \sum_{i\in U} d_i + r_i) \in \mathbb{Z}_q^{\lambda/l}$$

and then sets

$$\mathbf{c}_j := \mathbf{G}^{-1} \cdot \mathbf{v} - \sum_{i\in U\backslash\{j\}} \mathbf{c}_i - \sum_{i\notin U} \mathbf{s}_i^T \cdot \mathbf{A}.$$

Note that we compute the *left* inverse of $\mathbf{G}$ such that $\mathbf{G}^{-1}\cdot\mathbf{G} = \mathbf{I}_{\lambda}$. Finally, $\mathcal{B}$ sends all $\{\mathbf{c}_i\}^{i\in U}$ to $\mathcal{A}$.

Guess. If $\mathcal{A}$ has more than negligible advantage in winning the aggregator obliviousness security game, she can distinguish the ciphertexts from random. Specifically, if $\mathbf{c}_k = \mathbf{b}^T$ is indeed a valid $\mathsf{A\text{-}LWE}$ sample it is a valid encryption of (d_k, r_k) and $\mathcal{A}$ will return 0, otherwise she returns 1. Therefore, by forwarding A's output to $\mathcal{C}$ as her guess, $\mathcal{B}$ wins the game: she can distinguish $\mathbf{b}^T$ from random and solve decision $\mathsf{A\text{-}LWE}^{f}_{\kappa,\lambda,\sigma}$. ∎

Correctness of Adapted BGV: Following the parameter analysis in [19] and [9], we first analyze the expected magnitudes of values sampled from the distributions listed above Definition 11 before estimating the noise generated in each part of the scheme.

For our particular ring setting where $R = \mathbb{Z}[X]/\Phi_m(X)$ and is set as a power of 2 we bound the p-norm of a ring

element $x \in R$ using its canonical embedding $\mathsf{can}(x) : R \to \mathbb{C}^{\phi(m)}$, i.e. $||x||_\infty \leq ||x||_\infty^{\mathsf{can}} \leq ||x||_1$, where $||x||_\infty^{\mathsf{can}} = ||\mathsf{can}(x)||_\infty$. can maps a ring element x to a $\phi(m)$-vector where each coefficient is an evaluation of x on the complex primitive m-th root of unity ζ_m^i over all $i \in (\mathbb{Z}/m\mathbb{Z})^*$.

Sampling $x \in R$ from $\mathcal{ZO}(0.5, \phi(m))$ generates a random variable with variance $\mathsf{Var}_Z = \frac{1}{2}\phi(m)$. With distribution $\mathcal{DG}(\sigma'^2, \phi(m))$ we get $\mathsf{Var}_G = \sigma'^2 \cdot \phi(m)$ and $\mathcal{U}(q, \phi(m))$ yields $\mathsf{Var}_U = \frac{q^2}{12} \cdot \phi(m)$.

By the law of large numbers, $||x||_\infty^{\mathsf{can}}$ is bounded by $6 \cdot \sqrt{\mathsf{Var}_i}$ w.h.p., since $\mathsf{erfc}(6) \approx 2^{-55}$, where $i \in \{Z, G, U\}$ depending on which distribution x is sampled from. For two such elements $x, y \in R$ with variances $\mathsf{Var}(\mathsf{can}(x))$ and $\mathsf{Var}(\mathsf{can}(y))$ respectively, we bound the product $||x \cdot y||_\infty^{\mathsf{can}}$ by $16 \cdot \sqrt{\mathsf{Var}(\mathsf{can}(x)) \cdot \mathsf{Var}(\mathsf{can}(y))}$, since $\mathsf{erfc}(4)^2 \approx 2^{-50}$. Consequently, we get the following bounds on the secret key s and the public key components a and ϵ from the key generation routine BGV.Gen according to Definition 11:

$$\begin{aligned}\mathsf{Var}(\mathsf{can}(a)) &= \mathsf{Var}_U = \frac{q_1^2}{12} \cdot \phi(m)\\ \mathsf{Var}(\mathsf{can}(s)) &= \mathsf{Var}_G = \sigma'^2 \cdot \phi(m)\\ \mathsf{Var}(\mathsf{can}(\epsilon)) &= \mathsf{Var}_G = \sigma'^2 \cdot \phi(m).\end{aligned}$$

As in [9] we define the noise of a ciphertext $c = (c_0, c_1)$ as an upper bound on $||c_0 - s \cdot c_1||_\infty^{\mathsf{can}}$. In the following we look at the noise from "fresh" ciphertexts, i.e. those generated by BGV.Enc, and the noise in reduced ciphertexts, i.e. outputs of SwitchModulus that is invoked during decryption in BGV.Dec bounded according to Definition 11.

Fresh ciphertexts. If $c = (c_0, c_1) = \mathsf{BGV.Enc}(pk, \mu)$ then the noise in c is bounded by

$$\begin{aligned}&||c_0 - s \cdot c_1||_\infty \leq ||c_0 - s \cdot c_1||_\infty^{\mathsf{can}}\\ &= ||(a \cdot s + p \cdot \epsilon) \cdot v + p \cdot e_0 + \mu - s \cdot (a \cdot v + p \cdot e_1)||_\infty^{\mathsf{can}}\\ &= ||\mu + p \cdot (\epsilon \cdot v + e_0 - e_1 \cdot s)||_\infty^{\mathsf{can}}\\ &\leq ||\mu||_\infty^{\mathsf{can}} + p \cdot (||\epsilon \cdot v||_\infty^{\mathsf{can}} + ||e_0||_\infty^{\mathsf{can}} + ||e_1 \cdot s||_\infty^{\mathsf{can}})\\ &\leq \phi(m) \cdot (p-1) + p \cdot \Big(16 \cdot \sqrt{\sigma'^2 \cdot \phi(m) \cdot \frac{1}{2} \cdot \phi(m)} +\\ &\quad 6 \cdot \sqrt{\sigma'^2 \cdot \phi(m)} + 16 \cdot \sqrt{\sigma'^2 \cdot \phi(m) \cdot \sigma'^2 \cdot \phi(m)}\Big)\\ &= \phi(m) \cdot (p-1) + 2p\sigma' \cdot ((8 + 4\sqrt{2}) \cdot \phi(m) + 3 \cdot \sqrt{\phi(m)})\\ &\quad = B_{clean}.\end{aligned}$$

Reduced ciphertexts. If input ciphertext c has noise ν then output ciphertext $c' = \mathsf{SwitchModulus}(\mathsf{c_0}, \mathsf{c_1})$ has noise ν' where $\nu' = \frac{q_0}{q_1} \cdot \nu + B_{scale} = \frac{\nu}{p_1} + B_{scale}$. Recall that $q_0 = p_0 \cdot p_1$. B_{scale} captures overhead noise from the rounding error caused by reducing to modulus $q_1 = p_1$. Let $\tau = (\tau_0, \tau_1)$ be the rounding error, i.e. $(\tau_0, \tau_1) = (c_0', c_1') - \frac{q_0}{q_1}(c_0, c_1)$. Then, $\mathsf{can}(\tau_i)$ is roughly distributed according to a complex Gaussian with variance $\frac{p^2}{12} \cdot \phi(m)$. Therefore,

$$||\tau_0 + \tau_1 \cdot s||_\infty^{\mathsf{can}} \quad \leq \quad \frac{1}{\sqrt{3}} \cdot p \cdot (3 \cdot \sqrt{\phi(m)} + \sigma' \cdot \phi(m)) = B_{scale}.$$

Sum of ciphertexts. Summing ciphertexts $c_1, \ldots, c_N$ with noises $\nu_1, \ldots, \nu_N$ respectively, results in overall noise $\nu = \sum_{i=1}^N \nu_i$.

BGV.Dec takes a sum of some N ciphertexts $c = \sum_{i=1}^N c_i$ as input, where each $c_i \leftarrow \mathsf{BGV.Enc}$ is a fresh ciphertext. Then c is reduced to c' using $\mathsf{SwitchModulus}(c, q_1, q_0)$ and the plaintext is retrieved via $(c_0' - s \cdot c_1' \mod q_0) \mod p$. Therefore, in order to decrypt correctly $\nu' < \frac{q_0}{2} = \frac{p_0}{2}$, where ν' is the noise associated to c'. We can bound ν' using the bounds given above:

$$\begin{aligned}\nu' &\leq \frac{N \cdot B_{clean}}{p_1} + B_{scale}\\ &= \frac{N \cdot \Big(\phi(m) \cdot (p-1) + 2p\sigma' \cdot ((8 + 4\sqrt{2}) \cdot \phi(m) + 3 \cdot \sqrt{\phi(m)})\Big)}{p_1}\\ &\quad + \frac{1}{\sqrt{3}} \cdot p \cdot (3 \cdot \sqrt{\phi(m)} + \sigma' \cdot \phi(m)).\end{aligned}$$