PPSA: Polynomial Private Stream Aggregation for Time-Series Data Analysis

Antonia Januszewicz[1], Daniela Medrano Gutiérrez[2], Nirajan Koirala[1], Jiachen Zhao[1], Jonathan Takeshita[4], Jaewoo Lee[3], and Taeho Jung[1]

[1]University of Notre Dame, Notre Dame, IN 46556, USA
[2]Universidad Panamericana, Mexico City, [3]University of Georgia, Athens, [4]Tokyo Institute of Technology, Meguro, Tokyo, Japan
[1]{ajanusze,nkoirala,jzhao7,tjung}nd.edu,[2]dmedgutz@gmail.com,[4]jtakeshi@nd.edu,
[3]jl91659@uga.edu

Abstract. Modern data analytics requires computing functions on streams of data points from many users that are challenging to calculate, due to both the high scale and nontrivial nature of the computation at hand. The need for data privacy complicates this matter further, as general-purpose privacy-enhancing technologies face limitations in at least scalability or utility. Existing work has attempted to improve this by designing purpose-built protocols for the use case of Private Stream Aggregation; however, prior work lacks the ability to compute more general aggregative functions without the assumption of trusted parties or hardware.
In this work, we present PPSA, a protocol that performs Private Polynomial Stream Aggregation, allowing the private computation of any polynomial function on user data streams even in the presence of an untrusted aggregator. Unlike previous state-of-the-art approaches, PPSA enables secure aggregation beyond simple summations without relying on trusted hardware; it utilizes only tools from cryptography and differential privacy. Our experiments show that PPSA has low latency during the encryption and aggregation processes with an encryption latency of 10.5 ms and aggregation latency of 21.6 ms for 1000 users, which are up to 138× faster than the state-of-the-art prior work.

Keywords: Public key cryptosystems · Lattice-based Cryptography, · Private Stream Aggregation.

1 Introduction

Private Stream Aggregation (PSA) protocols allow aggregative computations on time-series user data. It is mostly focused on the summation of data, which is not sufficient for more complex data analysis such as regressions or modeling. What work does exist in polynomial PSA, and particularly post-quantum PSA, requires the addition of specific hardware or higher communication overhead. Our approach uses current solutions for post-quantum secure aggregation but applies the Complex Canonical Embedding (CCE) commonly used in Fully Homomorphic Encryption (FHE) to transform the input in a way that allows us to extract the product efficiently and securely. As such, our approach produces polynomial aggregation without requiring additional outside resources.

Background and Motivation The core idea of PSA [37] is the need for a research method such that data can be gathered on sensitive subjects without the leakage of any one person's information. For example, hospital patients should never worry that information on their cancer diagnosis is shared with the world, but medical researchers still need patient information to be able to research how to cure cancer. At the heart of this contradiction lies the tradeoff between security and accuracy, accurately represented in PSA with the use of differential privacy (DP). PSA protocols employ algorithms to encrypt user data and sufficiently obscure personally identifiable traits to allow people to freely share their data, without worrying about a leak of their information throughout the data-gathering process either in data storage or by a malicious aggregator. PSA, a particular case of Secure Aggregation (SA), is a class of secure protocols built to collect data from distributed sources in a way that does not leak any one individual's data. It is widely used for many purposes, such as smart metering or federated learning, to ensure user privacy [45,44].

Limitations of State-of-the-art PSA For time-series data analysis, specifically PSA, the most challenging aspect is how to make algorithms with a low enough overhead to be used in real-life large-scale applications [25,40,44,41]. In the past year, that challenge has been compounded by the movement towards Post-Quantum secure algorithms, which increase computational complexity [10,19,39]. In particular, there seems to be a lack of research on how to account for fault tolerance and computational variability, particularly for the newest Post-Quantum secure algorithms. For data to be taken from a wide range of sources over time, it is necessary to permit a number of users to be unavailable during all rounds of data collection [18,28,45]. Additionally, the classic PSA only collects the sum of the data. As such, PSA may be insufficient for certain research that requires further computation, such as when the product of the provided data may be necessary. To do this, we present a new Post-Quantum PSA protocol that has both fault tolerance and finds the product of provided user data, and provide both theoretical and practical proof of its efficacy through theory and model testing. We wish to investigate how our protocol compares with existing protocols in terms of speed and accuracy.

Research Challenges The primary challenge is the difficulty of supporting multiplicative aggregation in Post-Quantum PSA due to the difficulty of supporting high multiplicative depths in lattice-based cryptography [4]. Due to this challenge, it is highly nontrivial to expand the functionality of Post-Quantum PSA beyond simple summation.

Our Work In this work, we present the first-of-its-kind polynomial PSA protocol with quantum security and minimal overhead. Our protocol, PPSA: Polynomial PSA, is practical for collecting data for computationally complex analysis like data modeling or regressions. The novelty of our approach lies in the way differential privacy and complex canonical embedding are used to achieve polynomial aggregation.

Our Contributions

1. We present PPSA, the first RLWE-based PSA scheme that can be used to perform polynomial computations (instead of just summation) on user data without relying on trusted hardware. This is enabled by a novel idea that intelligently combines PSA and differential privacy.
2. We propose extensions of PPSA for practical issues such as fault tolerance and differential privacy with parallel composition.
3. We implement PPSA and provide source codes for reproducibility and extensions. The extensive experimental results with real implementations demonstrates that its performance is superior to a state-of-the-art prior polynomial aggregation protocol relying on Trusted Execution Environment.
   For $n = 1000$ users and a plaintext space of $|t| = 16$ bits, PPSA encryption achieves a latency of only 10.5 ms, and aggregation and decryption run in 21.6 ms. Our experiments with increasing numbers of users shows that PPSA is practically scalable for real-world deployments.

2 Related Work

There are three important aspects of PSA protocols: whether they are Post-Quantum or fault-tolerant and what functions they support. As it stands, most protocols were created before the need for Post-Quantum cryptography arose, and so do not address the issue. In particular, fault tolerance is also thoroughly addressed in many protocols, but mostly ones that are not Post-Quantum. A smaller category of protocols concerns protocols that can handle more than the summation of data, in particular the product of data. As such, the clear current gap in knowledge can be found at the intersection of these three categories. The presence of trusted third parties fluctuates in different protocols, but there is insufficient research on whether the presence of such a third party can be entirely avoided.

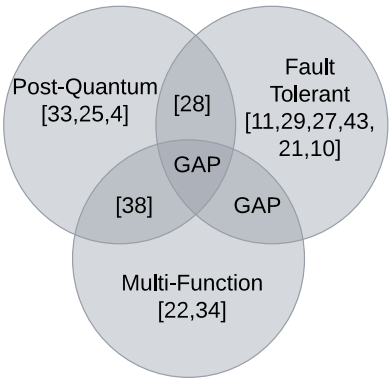

Fig. 1: PSA Protocols [35,26,4,29,40,23,24,11,30,28,45,21,10]
Not Post-Quantum secure, fault tolerant, or support multiplication: [18,5,6,37].

2.1 Post-Quantum Private Stream Aggregation

Post-Quantum versions of PSA usually fall into three categories when it comes to general methods or techniques: Secure Multi-Party Computation (SMC), Trusted Hardware, or Homomorphic Encryption (HE). These tools can be used as a black box in the code by relying on a direct transformation, or by examining and replicating underlying techniques such as secret sharing.

HE is a type of cryptography that allows for operations to be done on ciphertext that reflect in the plaintext after decryption [8]. SMC can achieve lower computational overhead than HE, but using it both directly or using secret sharing accrues more communication overhead [6]. Trusted Hardware also has lower computational overhead, but as it requires a trusted physical implementation, it is the least popular of the three options [26]. As SMC and Trusted Execution Environments (TEE) are not affected by Shor's algorithm, only protocols that used HE were affected by its discovery [39]. However, these schemes often have much higher overhead than that of the prior protocols. The lack in recent solutions is that the overhead is not low enough for the protocols to be able to compete with non-Post-Quantum secure protocols [38,19,4].

Examples of security assumptions that are not Post-Quantum secure and appear most commonly in PSA protocols include Diffie-Hellman, LEOM, and the Decisional Composite Residuosity Assumption (DCRA) [28,21,18]. Modern HE utilizes lattice-based cryptography, often relying on the Learning with Errors (LWE) Ring Learning With Errors (RLWE) assumptions [33,9,2]. These assumptions allowed for PSA to be implemented with the use of Fully Homomorphic Encryption (FHE). Two encryption algorithms that serve as inspiration in particular in later PSA are the BGV and CKKS FHE schemes [7,14]. There are some other FHE schemes that, while not initially Post-Quantum secure can be made Post-Quantum secure with the addition of elliptic curve cryptography to places where the Diffie-Hellman Key exchange is used [31].

2.2 Fault Tolerance

The classic version of PSA presented in Shi et. al is not fault-tolerant because it was only designed for one round of data collection, but soon after there many were developed protocols that sought to rectify that with the use of periodic distributive SA, which endlessly exchanged information [37,11]. Protocols that do not address fault tolerance usually require the keys and values to all be recalculated, accruing significant overhead. There are two ways to handle failures, reactively and proactively. To handle failures reactively, the protocol can group the client nodes in a certain way or by changing the encryption to be able to handle the lack of an input from some number of clients [30,28]. To handle failures proactively, clients can send filler values in advance which can be drawn from in the case that client cannot send their data on time or a trusted third party can detect and send in the dummy data itself [45,13]. Another possible

fix for fault tolerance with only slight modifications to the protocol itself is to add trusted hardware, but that does add further technology requirements on the part of the aggregator [26].

2.3 Beyond Sums

Although PSA is usually tailored for summation of data, there are several algorithms that have emerged that can be also used in multiplication [23,24,27]. However, to construct these algorithms, there is a trade-off in terms of the overhead. In the case of the algorithms which are not Post-Quantum secure, the trade-off is in terms of communication complexity. These algorithms require clients to send their information to a number of others in their direct neighborhood unlike regular FHE PSA, which only requires one message between aggregator and client [23,24,27].

2.4 Trusted Third Party

One question, that can determine the usability of some PSA protocols in real-life situations, is the need for a trusted third party. Some prior PSA schemes require an entity to be a trusted key dealer, a party that calculates the secret keys of the clients and aggregator and sends them out to all parties [39]. In this case, the key-dealer is commonly used because of the requirement that the secret keys of the users be used to create the aggregator secret key [29]. Without such a key dealer, the usual way for the users to collaborate and create secret keys which correspond to the aggregator secret key is if there is inter-client communication, which is often not possible due to the nature of the application and introduces additional communication overhead [12]. It has been noted that an untrusted third party can also be used to facilitate (PRE) Proxy Re-Encryption to assist with fault tolerance and reduce the need for a trusted key dealer, but there has not yet been a Post-Quantum version of this protocol [35].

3 Background

3.1 Private Stream Aggregation (PSA)

In the scenario of PSA, we assume that there are n users, each of whom has a stream of time-series data. The users will continually send messages to the aggregator, who is honest-but-curious, i.e., they will execute the protocol correctly, but may also carry out additional computations to try to learn about user data. A PSA scheme allows the aggregator to learn an aggregative function (i.e., sum, product) of all user data points emitted at the same timestamp, and ideally nothing else.
The formal definition of PSA schemes includes the following three algorithms [39,37,4]:

– *Setup*: The user keys and other values (e.g., nonces) are distributed or generated along with the aggregator key.

– $NoisyEnc$: Each user uses this function and their individual key to create their own ciphertext that will be collected by the aggregator.
– $AggrDec$: The aggregator uses this function when it is in possession of all the ciphertexts to get the noisy sum of the user data.

Some algorithms may require additional functions or actions, but all PSA algorithms at least use these three algorithms during the course of the protocol. PSA usually guarantees *Aggregator Obliviousness*, which limits how much information the aggregator can gain from the information it receives [37]. Although PSA is usually applied for summation, protocols have emerged that can compute multiplication and other functions beyond simple additive aggregation [23,24,27].

3.2 Differential Privacy for PSA

PSA can use differential privacy (DP) to ensure the privacy of individual clients' data values. During the process of data aggregation, the aggregator will normally have access to all the data points that they wish to aggregate. Even though data can be anonymized and names stripped, there is always a possibility that the data contains sufficient personal information for their name or other identification to be extrapolated. To prevent data from being traced back to the source, DP introduces noise to obscure individual data while maintaining some level of accuracy across the whole set [17]. The privacy guarantee relies on the trade-off between the accuracy of the data and the identifiability of specific data points [16].

In particular, DP can usually be split into the two categories of Local or Global DP, each of which has different privacy assumptions. In global differential privacy (GDP), the assumption is that the aggregator is a trusted party, but the requester is not trusted. Local differential privacy (LDP) assumes that not even the aggregator is a trusted party, and may attempt to use their position to gain additional access to information, commonly referred to as an *honest but curious* party [36].

DP can be used in conjunction with many types of cryptography. When used in an encryption scheme, the addition of differentially private noise and encryption may be different steps [42]. Additionally, the DP parameters chosen in the implementation of a system reflect whether privacy or accuracy is the priority [15]. DP is commonly used in PSA because of the sensitive nature of the data [37,4,39]. During a chosen-plaintext attack, the querier or aggregator could request the sum of data for some n entities and then request the same sum for $n-1$ entities, which leads to the revelation of the information of that missing individual [17]. As encryption cannot be used to counter this attack, DP is commonly used in conjunction with encryption [37].

3.3 Residue Number System

Theorem 1. *The **Chinese Remainder Theorem** states that if $q_1, ..., q_k$ are pairwise co-prime integers greater than 1, we define $q = \prod_{i=1}^{k} q_i$. Then for $x \in \mathbb{Z}_q$, the map*

$$x \ mod \ q \mapsto (x \ mod \ q_1, ...x \ mod \ q_k)$$

defines a ring isomorphism

$$\mathbb{Z}/q\mathbb{Z} \cong \mathbb{Z}/q_1\mathbb{Z} \times ... \times \mathbb{Z}/q_k\mathbb{Z}$$

We call the set of coprime factors a *Residue Number System (RNS) basis*. This is useful for arithmetic on numbers modulo a large value q. By choosing RNS moduli q_i to be less than a computer word (usually 64 bits on modern systems), representing a number $x \in \mathbb{Z}_q$ as the tuple $(x \bmod q_1, ...x \bmod q_k)$ can avoid costly multiprecision integer arithmetic, greatly improving the efficiency of addition and multiplication on numbers modulo q [3,20].

3.4 The Number-Theoretic Transform (NTT)

The Number-Theoretic Transform (NTT), a variant of the Fast Fourier Transform, is useful for performing fast polynomial multiplication [32]. A textbook algorithm for polynomial multiplication on polynomials of degree N has a runtime of $\mathcal{O}(N^2)$. However, the NTT has a runtime of $\mathcal{O}(N \cdot log(N))$. Coefficientwise multiplication (an $\mathcal{O}(N)$ operation) on polynomials that have had the NTT applied to them corresponds to polynomial multiplication on the original polynomials. Thus, applying the NTT, performing coefficientwise multiplication, and then applying the inverse NTT to the result can accomplish polynomial multiplication in log-linear time.

Let $\mathbb{Z}_p$ be an integer ring modulo a prime p, and let $P(x)$ denote a polynomial in $\mathbb{Z}_p[x]$. Define ω as the primitive $N-th$ root of unity in $\mathbb{Z}_p$, and ψ as the primitive $N-th$ root of unity in $\mathbb{Z}_p$ such that $\omega^N \equiv 1 \pmod{\text{p}}$ and $\psi^2 \equiv \omega \pmod{\text{p}}$.
The Number-Theoretic Transform of the coefficients $\mathbf{a}$ of $P(x)$ is the map $\mathbb{Z}_p^N \mapsto \mathbb{Z}_p^N$ defined by:

$$\hat{a}_j = \sum_{i=0}^{N-1} \psi^{2ij+i} a_i \pmod{\text{p}}$$

Using the properties of ψ: $\psi^{k+2N} = \psi^k$ and $\psi^{k+N} = -\psi^k$, we obtain:

$$\hat{a}_j = \sum_{i=0}^{N-1} \psi^{2ij+i} a_i \pmod{\text{p}} = \sum_{i=0}^{\frac{N}{2}-1} \psi^{4ij+2i} a_{2i} + \psi^{2j+1} \sum_{i=0}^{\frac{N}{2}-1} \psi^{4ij+2i} a_{2i+1} \pmod{\text{p}}$$

$$\hat{a}_{j+N/2} = \sum_{i=0}^{\frac{N}{2}-1} \psi^{4ij+2i} a_{2i} - \psi^{2j+1} \sum_{i=0}^{\frac{N}{2}-1} \psi^{4ij+2i} a_{2i+1} \pmod{\text{p}}$$

3.5 The Complex Canonical Embedding (CCE)

The Complex Canonical Embedding comprises a map between vectors in $\mathbb{C}^{\frac{N}{2}}$ and polynomials in $R_q = \mathbb{Z}_q[X] \pmod{X^N + 1}$. A CCE σ is defined as $\sigma : R[X] \mapsto C^N$. Within the context of this paper, we are concerned with $\sigma : \mathbb{Z}_q[X]/(\Phi_M(X)) \mapsto C^N$ with $\Phi_M(X)$ denoting the M-th cyclotomic polynomial. Let $\mathcal{E}_M = e^{\frac{2i\pi}{M}}$ be the primitive M-th root of unity, the N roots of $\Phi_M(x)$ is the set $\mathcal{S} = \{\mathcal{E}_M, \mathcal{E}_M^3, ..., \mathcal{E}_M^{2N-1}\}$. For an element $P(x) \in \mathbb{Z}_q[X]/(\Phi_M(X))$, the CCE $\sigma : \mathbb{Z}_q[X]/(\Phi_M(X)) \mapsto C^N$ of $P(x)$ is defined as:

$$\sigma(P(x)) = (P(\mathcal{E}_M), P(\mathcal{E}_M^3), ...P(\mathcal{E}_M^{2N-1}))^T$$

3.6 Definition of Security

Definition 1. *A polynomial PSA scheme is differentially private aggregator oblivious in the encrypt-once model if any probabilistic polynomial-time (PPT) adversary has no more than negligible advantage with respect to security parameters $(\epsilon, \delta, \lambda)$ in the following security game:*

Setup *The challenger runs the Setup algorithm, returning any public parameters to the adversary.*

Queries *The adversary may make up to poly(λ) of following types of queries adaptively:*

- Encrypt: The adversary may specify $(i, j, ts, c_j, e_{i,j}, x_{i,j,ts}, r_{i,j,ts})$ and ask for the ciphertext. The challenger returns the ciphertext $c_{i,ts} = Enc(s_i, ts, e_{i,j}, x_{i,j,ts}, r_{i,j,ts})$ to the adversary.
- Compromise: The adversary specifies a party $i \in [0, n) \cup \{\square\}$. If $i = \square$, the challenger returns the aggregator's decryption key s' to the adversary (i.e., the aggregator is compromised.). Otherwise, the challenger returns user i's secret key s_i, to the adversary (i.e., user i is compromised).
- Challenge: This query is only made once. The adversary specifies a set of participants U and a time ts, such that neither ts nor any $i \in U$ was previously argued to Compromise. For each user $i \in U$, the adversary chooses a pair of input sets. If the aggregator was compromised, the adversary has further restrictions that polynomial outcomes from the two input sets are identical such that the adversary cannot infer from the outcome which input set was used in encryption. The challenger then randomly chooses one of the two input sets to generate and publish their ciphertexts.

Guess *The adversary attempts to guess which input set was encrypted. The adversary wins if they can guess the input set correctly.*

Aggregator obliviousness here means that the protocol execution won't leak any more data than what groups can collude to figure out [37].

4 PPSA Construction

In many scenarios, parties $i \in S$ holding time-series private data $m_{i,j,ts}$ wish to privately calculate a polynomial function $\sum_{j=0}^{d} c_j \prod_{i \in S} m_{i,j,ts}^{e_{i,j}}$ for publicly known values $c_j, e_{i,j}$ at a particular timestamp ts. We can achieve this with the combinations of semantic security on intermediate calculations and differential privacy for the outcome. Our idea is to perform the calculations of product terms separately with the inclusion of differential privacy and then to perform the final summation of polynomial terms.

4.1 Basic Construction

Formally, we define our protocol as follows. Suppose S contains n users, and that a Private Stream Aggregation PSA scheme $PSA = (Setup, Enc, Agg)$ exists. Our polynomial aggregation scheme $PPSA$ can then be defined with the following procedures:

- $PPSA.Setup$: Given the number of users n, the desired levels of differential privacy and accuracy for each coefficient of the polynomial, and the desired level of semantic security for ciphertexts, use $PSA.Setup$ to derive a set of parameters $parms$ that satisfy the given constraints. $parms$ includes a Discrete Laplacian distribution DL_s parameterized to a scale parameter s [4].
- $PPSA.Enc$: Given a PSA secret key sk_i and a set of data $m_{i,j,ts}$ for some particular timestamp ts, set $x_{i,j,ts} = CCE(ln(m_{i,j,ts}^{e_{i,j}} + r_{i,j,ts}))$, where each $r_{i,j,ts}$ is drawn from DL_s and ln is the natural log. Then, return

$$CT_i = \{ct_{i,j,ts} = PSA.Enc(parms, x_{i,j,ts}, sk_i, ts)\}_{j \in [0,d]} \tag{1}$$

- $PPSA.Agg$: Given a PSA aggregator key sk', a timestamp ts, and the ciphertext sets $\{CT_i\}_{i \in S}$, first calculate the values $coeff_j = exp(CCE^{-1}(PSA.Agg(parms, \{ct_{i,j}\}_{i \in S}, sk', ts)))$ where exp is the exponential function. Here, $coeff_j$ are the polynomial coefficients which are then attached to the evaluation given by the vector c_j of the polynomial. (Note that each of these values is not exactly the product $\prod_{i \in S} m_{i,j,ts}^{e_{i,j}}$, due to the differentially private noise used.) Then, simply compute the result $y = \sum_{j \in [0,d]} c_j \cdot coeff_j$.

Correctness A correct PSA scheme will ensure the following:

$$PSA.Agg(parms, \{PSA.Enc(parms, x_{i,ts}, sk_i, ts\}_{i \in \mathbb{Z}_n}, sk', ts) = \sum_{i=0}^{n-1} x_{i,ts}$$

By this property, $coeff_j = exp\Big(CCE^{-1}\Big(\sum_{i=0}^{n-1} CCE\Big(ln\big(m_{i,j,ts}^{e_{i,j}} + r_{i,j,ts}\big)\Big)\Big)\Big) = exp\Big(\sum_{i=0}^{n-1} ln\big(m_{i,j,ts}^{e_{i,j}} + r_{i,j,ts}\big)\Big) = \prod_{i=0}^{n-1} m_{i,j,ts}^{e_{i,j}} + r_{i,j,ts} \approx \prod_{i=0}^{n-1} m_{i,j,ts}^{e_{i,j}}$. It then easily follows that multiplying each term $coeff_j$ yields the desired polynomial.

Table 1: Common Important Notation

Notation	Meaning
λ	Security parameter, measured in bits.
R_q	The polynomial ring that ciphertexts and other terms reside in.
q	Ciphertext modulus, an integer that is commonly hundreds of bits long.
N	The polynomial modulus degree, a power of two commonly ranging from 2^{10} to 2^{15}.
t	The plaintext modulus, an integer much less than q. Commonly ranges from 16 to 192.
n	Number of users in the PSA instantiation.
χ, ζ	Small error distributions on R_q.
ts	The timestamp at which a particular aggregation takes place.
$m_{i,j,ts}$	Time-series private data
$x_{i,j,ts}$	User i's PSA input at timestamp ts.
$r_{i,j,ts}$	User i's PSA noise j at timestamp ts, typically drawn from DL_s.
$e_{i,j,ts}$	User i's PSA error term at timestamp ts.
$ct_{i,j,ts}$	User i's ciphertext at timestamp ts.
sk_i	User i's PSA key
sk'	The aggregator's key.
H	A random-oracle hash function mapping from timestamps to R_q.
$parms$	The public PSA parameters R_q, t, n, H, and a discrete Laplacian distribution DL_s.
y_{ts}	The final aggregation result, equal to the sum of users' inputs and noise.
ϵ	Differential Privacy parameter related to privacy.
s, σ	Parameters of the Discrete Laplacian distribution DL_s.
k	Number of RNS moduli used for the ciphertext.
$q_0, q_1, \cdots, q_{k-1}$	Coprime ciphertext moduli used in RNS, whose product is q. For use in the NTT, these should each be equivalent to 1 modulo 2N.
$q_i^*, \tilde{q_i}, w_i, \theta_i$	Precomputed parameters used in RNS base conversion and division-with-rounding.
$k^{'}$	Number of RNS moduli used for the plaintext.

4.2 Preventing Repeated-Encryption Attacks

In the above basic protocol, there is a subtle flaw. It is easy to define the scheme in a way that is not vulnerable, but we explicitly show and fix the flaw in the interest of being informative.

As noted in previous work [43], creating multiple ciphertexts $ct_{i,j,ts}$ from a particular timestamp ts allows an attack. We thus modify the basic procedure to prevent this. Instead of setting $ct_{i,j,ts} = PSA.Enc(parms, x_{i,j,ts}^{e_{i,j}}, sk_i, ts)\}_{j\in[0,d]}$ (where every $ct_{i,j,ts}$ depends on ts exactly), we first set $ts_j = ts||j$. (Note that the size of timestamps is extended by $|n|$ bits.) Then, we can set $CT_{i,j,ts} = \{PSA.Enc(parms, x_{i,j,ts}^{e_{i,j}}, sk_i, ts_j)\}_{j\in[0,d]}$. This allows each user's invocations of $PSA.Enc$ to use a different timestamp ts_j. The timestamps should be similarly extended in $PPSA.Agg$.

5 Analysis

5.1 Security and Accuracy

Semantic security of the ciphertexts ct_{i,j,ts_j} follows directly from the semantic security of the underlying PSA scheme. We note that the values $coeff_j$ do not fully obscure user inputs; this sum of user data is protected only by differentially private noise. A corollary to this is that the values $coeff_j$ may differ from the actual sums $\sum_{i\in S} m_{i,j,ts}^{e_{i,j}}$, and thus the final result y may differ from the exact polynomial aggregate $\sum_{j=0}^{d} c_j \prod_{i\in S} m_{i,j,ts}^{e_{i,j}}$. A full analysis of the privacy and accuracy tradeoffs can be found in [39,4].

5.2 Achieving Differential Privacy

For this paper, we rely on the Discrete Laplacian distribution to add differentially private noise to the user inputs before aggregation, achieving the desired level of differential privacy.

Definition 2. *The **Discrete Laplacian** distribution is defined as following: for parameters $s > 1$, $\sigma = exp(-1/s) \in (0, 1)$, $x \in DL_s$ has probability mass function:*

$$DL_s(x) = \frac{1-\sigma}{1+\sigma} \cdot \sigma^{|x|}$$

We adopt the procedures of the Discrete Laplacian scheme from SLAP because it closely relates to the purpose of PPSA. For each time-series private data $m_{i,j,ts}$, we add a private noise $r_{i,j,ts}$. With probability β, $r_{i,j,ts}$ is drawn from a Discrete Laplacian distribution with parameter s and with probability $(1-\beta)$, $r_{i,j,ts}$ is zero.

Let the Discrete Laplacian parameter s to be $\frac{w}{\epsilon}$ determined by the desired level of privacy (ϵ, δ) and n users whose inputs fit within the width of w. With appropriate bounds on the proportion of honest users who would add differentially

private noise to their inputs ($\gamma \geq \frac{1}{n} ln(\frac{1}{\delta})$) and appropriate bounds on w ($w \geq \frac{\epsilon}{3}$), the individual noises drawn from DL_s can guarantee (ϵ, δ)-differential privacy and the scheme can achieve (α, β)-accuracy from [39], Theorem 1.

5.3 Security Guarantees

Theorem 2. *PPSA is a differentially private aggregator oblivious polynomial PSA scheme.*

We present an informal analysis only because the security guarantee follows trivially from existing state-of-the-art PSA schemes. Existing PSA schemes (e.g., [39]) provide the property of *aggregator obliviousness*, i.e., the aggregator cannot learn anything beyond the summation outcome, the differentially private noises added to each invocation of PSA scheme ensure an aggregator in the security game (Section 3.6) has no more than negligible advantage with respect to security parameters $(\epsilon, \delta, \lambda)$.

6 Extensions and Future Improvements

6.1 Fault Tolerance

We define the Lagrange coefficient for n users in the following manner. As described in [22], we define the Lagrange coefficient for $i \in \{1, \ldots, n\}$ and a set P of user IDs in $\{1, \ldots, n\}$ as $\mathcal{L}_{i,P}(x) = \prod_{j \in P \setminus \{i\}} \frac{x-j}{i-j}$, and use the simplified Lagrange coefficient $\mathcal{L}_{i,P}$ for a special case $\mathcal{L}_{i,P}(0)$.

In the Equation (1), s_i is the secret key of the user i. To achieve fault tolerance in this design, we replace this term s_i with some point on a polynomial. Now, if this polynomial is multiplied by a Lagrange coefficient, and the summation of these polynomials multiplied by the Lagrange coefficient becomes zero (or a multiple of q_{agg} where q_{agg} is the ciphertext modulus) and cancels out during the aggregation.

Polynomial Construction Suppose the aggregator uses n points to uniquely define a polynomial $q^d(x)$ of degree $d = n - 1$. The other coefficients of the polynomial are randomly chosen from a finite field $\mathbb{F}_{q_{agg}}$. The constant term of this polynomial is set to be zero (i.e., $q^d(0) = 0$).

During the key generation, each user i is provided with a set of polynomials $q^{(2)}(i), \ldots, q^{(n-1)}(i)$. Each key recipient i also receives one data point for each secret polynomial $q^d(i)$, where the polynomial $q^d(x)$ remains unknown to everyone because it is the sum of all users' random polynomials. However, specific points on this unknown polynomial can be jointly computed. Each key recipient i receives one of the shares, where the polynomial $q^d(x)$ remains unknown to

everyone and it can be computed by summing up all users' polynomials. Hence, using Lemma 6 in [22], for any set of users P, we have

$$\sum_{i \in P} q^{|P|-1}(i)\mathcal{L}_{i,P} = 0 \bmod q_{agg}.$$

During the aggregation, since the constant terms in all the polynomials are 0, the above sum equals $q^{|P|-1}(0) = 0 \bmod q_{agg}$.

Aggregation During the aggregation, the aggregator simply sums n ciphertexts $c_{i,ts}$ received from n users such that final aggregation value results in the sum of the users' inputs plus some noise $r_{i,ts}$ modulo plaintext modulus. To be able to support a dynamic number of total users (i.e., enable new users to join and old users to leave), we employ the following ideas:

Adding New Users Suppose we are adding a new user with ID $= n + 1$ in a group of n users $\{1, \ldots, n\}$. First, we need to update the key set (i.e., the random polynomial) of every user i in the old group $\{1, \ldots, n\}$. Then, each user i needs to acquire and add one extra polynomial item $q^n(i)$ into his/her key set. Aggregator provides all old users in the new group $\{1, \ldots, n + 1\}$ with a complete key set containing $q^2(i), \ldots, q^n(i)$. Secondly, the aggregator issues the new user $n + 1$ with his/her full key set $EK_{n+1} = \{q^{(2)}(n + 1), \ldots, q^{(n-1)}(n + 1), q^{(n)}(n + 1)\}$ such that they can participate in any future aggregation process.

Removing Existing Users During the departure of one or more users from the group, the scheme can ignore the existing users, and the keys of the rest of the users are not affected by this change.

6.2 Differential Privacy

One alternative method to achieve the desired differential privacy of our scheme is to utilize Parallel Composition. Parallel Composition is based on splitting the input data into disjoint subsets and performing a differentially private scheme on each subset. The scheme is as private as the worst-performing subset, as proven in Theorem 4 in [34]. Therefore, for the overall scheme to achieve ϵ-differential privacy, we just need to guarantee ϵ-differential privacy for each subset. However, the Parallel Composition scheme does achieve the result at the cost of accuracy and thus needs to be cautiously applied.

7 Experimental Evaluation

7.1 Testing scaling and overhead

We evaluated our proposed method for polynomial aggregation to observe its performance in various settings. Our source code files are available at https:

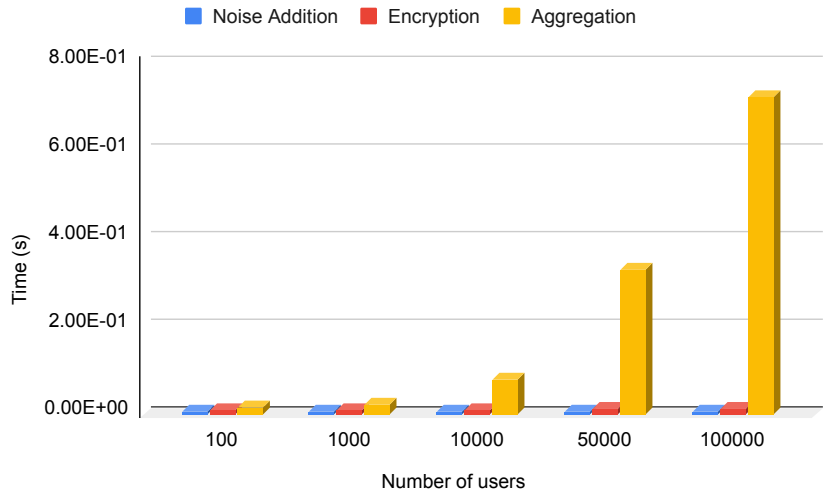

Fig. 2: Performance of our polynomial aggregation method for various functions

//github.com/cornflower26/PSAapp. For the implementation, we used C++17 and the OpenFHE library [1] as the PSA backend, which employs optimizations including RNS, SIMD encoding, and NTT. We used a security parameter $\lambda = 128$, a polynomial modulus degree $N = 1024$, and a ciphertext modulus $q = 64$ bits. In all our experiments, the plaintext modulus with the number of bits set to $|t| = 16$, except in cases where the number of users reaches 10^6, where $|t| = 10$ due to memory constraints on our machine when handling a large number of users. Our experiments were run on a server with an AMD EPYC 7313P processor and 128GB of memory, running Ubuntu 20.04. We evaluated the runtime for aggregation for different numbers of users ranging from 10^2 to 10^6 by performing 10 trials and reporting the average latency.

We computed polynomials of the form $\sum_{j=1}^{2} \prod_{i=1-j+1}^{n-j+1} m_{i,j,ts} + r_{i,j,ts}$ and report the timing data for adding differential privacy noise, encryption, and aggregation. Encryption time measures the time taken to encrypt one data input, while aggregation and decryption take into account the amount of users whose data needs to be aggregated. We can observe no increase in the time taken to encrypt a single data input when the number of users is increased. The aggregation time includes the decryption time as well, however, decryption is a small portion of the total aggregation time and does not significantly impact the aggregation timing. We report how the performance of our aggregation method changes as we increase the number of users in Figure 2. The pre-processing time, including the differential privacy noise addition and encryption time, remains under 0.0112 seconds for users up to 10^6. The aggregation time grows very slowly with the number of users and demonstrates very practical runtime for large-scale data aggregation applications. We note that the timing charts do not account for the network latencies of the users.

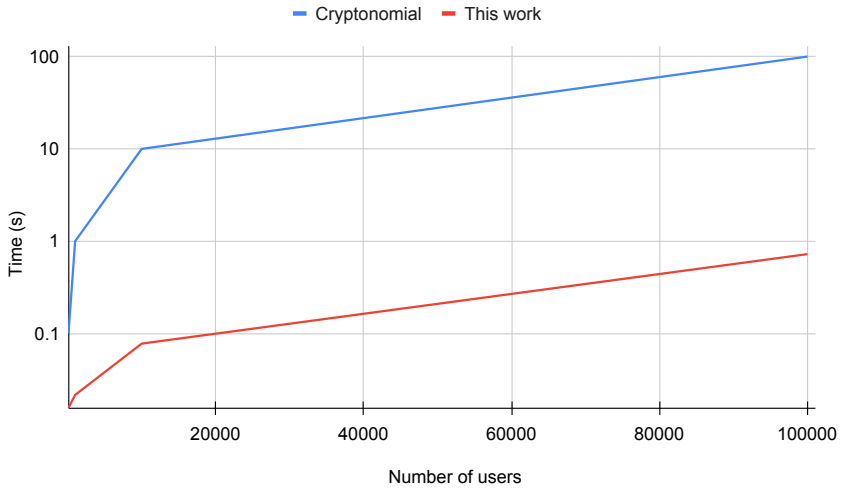

Fig. 3: Aggregation runtime for Cryptonomial [27] compared to PPSA

Comparison with prior work. Cryptonomial [27] is a state-of-the-art secure computation protocol proposed by Karl et al. for computing a multivariate polynomial that employs a Trusted Execution Environment. We used the experimental data provided in their work as the configurations of the machine they employed for their experiments closely resemble ours. We compare the aggregation latency of our method with Cryptonomial in Figure 3. Our method is almost $6 - 138\times$ faster than Cryptonomial for aggregating inputs for up to 10^6 users as we do not incur any overheads resulting from using TEEs. As there is no need for the data to be transferred in and out of the TEE enclave, which incurs additional overhead in Cryptonomial, the aggregation latency of our method is significantly lower, which results in faster performance.

7.2 Testing differential privacy and accuracy

In the following experiment, we will investigate a scenario involving a group of users and their data inputs, which are constrained within a specific interval. We aim is to explore the trade-offs between privacy guarantees and data accuracy.

We consider a scenario with $n = 100$ users, whose inputs fit in an interval of width w. In our experimental evaluations, we consider the parameters $\epsilon = 1$ and $\delta = 0.1$, which gives a minimum required proportion of honest participants $\gamma \geq \frac{1}{n}\ln(\frac{1}{\delta}) = 0.023$ in order to achieve (ϵ, δ)-differential privacy with values drawn from the discrete Laplacian with parameter $s = \frac{w}{\epsilon}$ and $w \geq \frac{\epsilon}{3}$ (see [39,4]). Furthermore, choosing $\beta = 0.05 \geq (20)^{-\frac{1}{0.023}} = (\frac{2}{\delta})^{-\frac{1}{\gamma}}$ gives us (α, β)-accuracy with $\alpha = \frac{4w}{\epsilon}\sqrt{\frac{1}{\gamma}\ln(\frac{1}{\delta})\ln(\frac{2}{\beta})} = 4w\sqrt{\frac{\ln(10)\ln(40)}{0.023}} = 4w\sqrt{\frac{\ln(10)\ln(40)}{0.023}} = 76.86w$. As we are testing with different means of user inputs and, consequently, different widths, we provide a table for the sake of completeness (see table 2).

Mean	Width (w)	Discrete Laplacian parameter (s)	α
100	1	1	76.86
1000	10	10	768.69
10000	100	100	7686.9
100000	1000	1000	76869

Table 2: Description of parameters when $\epsilon = 1$.

As the reader may observe in table 3 and figure 4, for each case, the minimum value is 0 and the maximum is 1. As the mean increases, the ratio consistently approaches a value very close to 1, indicating that the discrepancy between the actual polynomial aggregation and the polynomial aggregation with discrete Laplacian noise added diminishes.

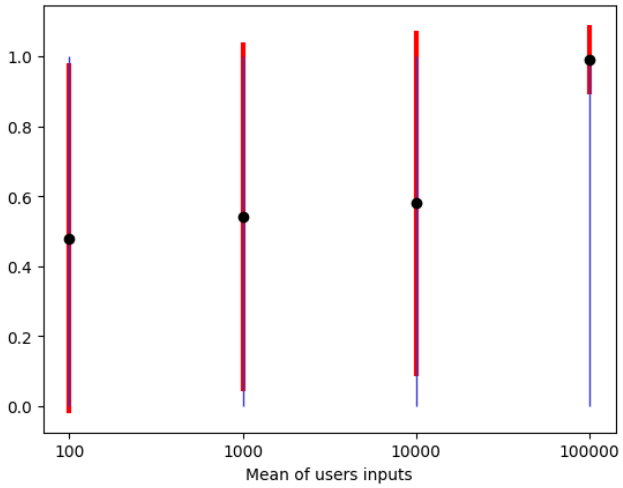

Fig. 4: Minimum (bottom of blue line), maximum (top of blue line), mean (dot) and standard deviation (red line) of the ratios of 100 samples when $\epsilon = 1$.

Now, we consider ϵ to be variable while $\delta = 0.01$ and mean $= 100$ are fixed. We still need the minimum required proportion of honest participants $\gamma \geq \frac{1}{n}\ln(\frac{1}{\delta}) = 0.023$ in order to achieve (ϵ, δ)-differential privacy with values drawn from the discrete Laplacian with parameter $s = \frac{w}{\epsilon}$ and $w \geq \frac{\epsilon}{3}$ (see [39,4]). Similarly, choosing $\beta = 0.05 \geq (20)^{-\frac{1}{0.023}} = (\frac{2}{\delta})^{-\frac{1}{\gamma}}$ gives us (α, β)-accuracy with $\alpha = \frac{4w}{\epsilon}\sqrt{\frac{1}{\gamma}\ln(\frac{1}{\delta})\ln(\frac{2}{\beta})} = \frac{4w}{\epsilon}\sqrt{\frac{\ln(10)\ln(40)}{0.023}}$. In table 4 we give the complete description of the parameters.

Mean of users inputs	Minimum	Maximum	Mean	Standard deviation
100	0	1	0.48	0.49959984
1000	0	1	0.54	0.49839743
10000	0	1	0.58	0.49355851
100000	0	1	0.99	0.09949874

Table 3: Minimum, maximum, mean and standard deviation of the ratios of 100 samples when $\epsilon = 1$.

Epsilon (ϵ)	Width (w)	Discrete Laplacian parameter (s)	α
0.01	0.01	1	76.86
0.1	0.1	1	76.86
1	1	1	76.86
10	10	1	76.86

Table 4: Description of parameters when mean = 100.

As the reader may observe in table 5 and figure 5, for each case, the minimum value is 0 and the maximum is 1. A minimum value of 0 suggests a significant difference between the polynomial aggregation and the polynomial aggregation with added noise. However, the crucial point is that the mean of the ratios, in each case, is nearly equal to 1, indicating the similarity between the calculated polynomials. Furthermore, we observe that the standard deviation is always lower than 0.2, implying that the ratios will almost certainly cluster around 1.

Epsilon (ϵ)	Minimum	Maximum	Mean	Standard deviation
0.01	0	1	0.99	0.09949874
0.1	0	1	0.98	0.14
1	0	1	0.99	0.09949874
10	0	1	0.96	0.19595918

Table 5: Minimum, maximum, mean and standard deviation of the ratios of 100 samples when mean = 100.

The code for the experiment previously described is available on GitHub at https://github.com/medranocode/PPSADataAnalysis.git.

8 Conclusion

In this paper, we presented PPSA, which builds on existing work in Private Stream Aggregation to implement private polynomial aggregation without the assumption of either a trusted party or trusted hardware. PPSA uses differential

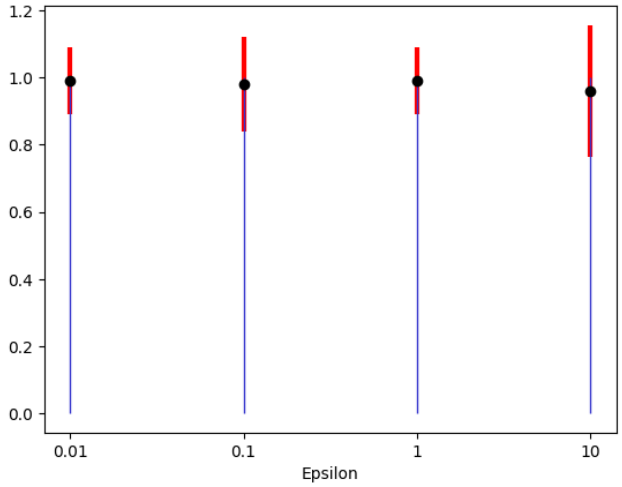

Fig. 5: Minimum (bottom of blue line), maximum (top of blue line), mean (dot) and standard deviation (red line) of the ratios of 100 samples when mean = 100.

privacy to protect intermediate values from an untrusted aggregator. Our experimental results show an aggregation latency of under 21.6 ms for 1000 users and a 16-bit plaintext space, with improvements over prior polynomial PSA with or without trusted hardware. The performance comparison between our PPSA protocol and Cryptonomial shows that there is the possibility for real-world deployment on any system, including one where trusted hardware is not available.

Acknowledgements

This material is based upon work supported by the Industry-University Cooperative Research Center Program at the US National Science Foundation under Grant No. 2224985.

References

1. A. Al Badawi, J. Bates, F. Bergamaschi, D. B. Cousins, S. Erabelli, N. Genise, S. Halevi, H. Hunt, A. Kim, Y. Lee, et al. Openfhe: Open-source fully homomorphic encryption library. In *Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, pages 53–63, 2022. 14
2. M. R. Albrecht, R. Player, and S. Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015. 4
3. J.-C. Bajard, J. Eynard, M. A. Hasan, and V. Zucca. A full rns variant of fv like somewhat homomorphic encryption schemes. In *International Conference on Selected Areas in Cryptography*, pages 423–442. Springer, 2016. 7
4. D. Becker, J. Guajardo, and K.-H. Zimmermann. Revisiting private stream aggregation: Lattice-based psa. In *NDSS*, volume 2, page 5, 2018. 2, 3, 4, 5, 6, 9, 11, 15, 16
5. F. Benhamouda, M. Joye, and B. Libert. A new framework for privacy-preserving aggregation of time-series data. *ACM Transactions on Information and System Security (TISSEC)*, 18(3):1–21, 2016. 3
6. K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017. 3, 4
7. Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014. 4
8. Z. Brakerski and V. Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *Annual cryptology conference*, pages 505–524. Springer, 2011. 4
9. Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. *SIAM Journal on computing*, 43(2):831–871, 2014. 4
10. J. Brorsson and M. Gunnarsson. Dipsauce: efficient private stream aggregation without trusted parties. In *Nordic Conference on Secure IT Systems*, pages 204–222. Springer, 2023. 2, 3
11. T. H. H. Chan, E. Shi, and D. Song. Privacy-preserving stream aggregation with fault tolerance. In *Financial Cryptography and Data Security: 16th International Conference, FC 2012, Kralendijk, Bonaire, Februray 27-March 2, 2012, Revised Selected Papers 16*, pages 200–214. Springer, 2012. 3, 4
12. J. Chen, H. Ma, and D. Zhao. Private data aggregation with integrity assurance and fault tolerance for mobile crowd-sensing. *Wireless Networks*, 23:131–144, 2017. 5
13. L. Chen, R. Lu, and Z. Cao. Pdaft: A privacy-preserving data aggregation scheme with fault tolerance for smart grid communications. *Peer-to-Peer networking and applications*, 8:1122–1132, 2015. 4
14. J. H. Cheon, A. Kim, M. Kim, and Y. Song. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23*, pages 409–437. Springer, 2017. 4
15. J. C. Duchi, M. I. Jordan, and M. J. Wainwright. Minimax optimal procedures for locally private estimation. *Journal of the American Statistical Association*, 113(521):182–201, 2018. 6

16. C. Dwork. Differential privacy. In *International colloquium on automata, languages, and programming*, pages 1–12. Springer, 2006. 6
17. C. Dwork, A. Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014. 6
18. K. Emura. Privacy-preserving aggregation of time-series data with public verifiability from simple assumptions. In *Information Security and Privacy: 22nd Australasian Conference, ACISP 2017, Auckland, New Zealand, July 3–5, 2017, Proceedings, Part II 22*, pages 193–213. Springer, 2017. 2, 3, 4
19. J. Ernst and A. Koch. Private stream aggregation with labels in the standard model. *Proceedings on Privacy Enhancing Technologies*, 2021. 2, 4
20. S. Halevi, Y. Polyakov, and V. Shoup. An improved rns variant of the bfv homomorphic encryption scheme. In *Topics in Cryptology–CT-RSA 2019: The Cryptographers' Track at the RSA Conference 2019, San Francisco, CA, USA, March 4–8, 2019, Proceedings*, pages 83–105. Springer, 2019. 7
21. M. Joye and B. Libert. A scalable scheme for privacy-preserving aggregation of time-series data. In *Financial Cryptography and Data Security: 17th International Conference, FC 2013, Okinawa, Japan, April 1-5, 2013, Revised Selected Papers 17*, pages 111–125. Springer, 2013. 3, 4
22. T. Jung, J. Han, and X.-Y. Li. Pda: semantically secure time-series data analytics with dynamic user groups. *IEEE Transactions on Dependable and Secure Computing*, 15(2):260–274, 2016. 12, 13
23. T. Jung, X.-Y. Li, and M. Wan. Collusion-tolerable privacy-preserving sum and product calculation without secure channel. *IEEE Transactions on Dependable and secure computing*, 12(1):45–57, 2014. 3, 5, 6
24. 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 *2013 Proceedings IEEE INFOCOM*, pages 2634–2642. IEEE, 2013. 3, 5, 6
25. P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al. Advances and open problems in federated learning. *Foundations and trends® in machine learning*, 14(1–2):1–210, 2021. 2
26. R. Karl, J. Takeshita, and T. Jung. Cryptonite: A framework for flexible time-series secure aggregation with non-interactive fault recovery. In *Security and Privacy in Communication Networks: 17th EAI International Conference, SecureComm 2021, Virtual Event, September 6–9, 2021, Proceedings, Part I 17*, pages 311–331. Springer, 2021. 3, 4, 5
27. R. Karl, J. Takeshita, A. Mohammed, A. Striegel, and T. Jung. Cryptonomial: a framework for private time-series polynomial calculations. In *Security and Privacy in Communication Networks: 17th EAI International Conference, SecureComm 2021, Virtual Event, September 6–9, 2021, Proceedings, Part I 17*, pages 332–351. Springer, 2021. 5, 6, 15
28. I. Leontiadis, K. Elkhiyaoui, and R. Molva. Private and dynamic time-series data aggregation with trust relaxation. In *Cryptology and Network Security: 13th International Conference, CANS 2014, Heraklion, Crete, Greece, October 22-24, 2014. Proceedings 13*, pages 305–320. Springer, 2014. 2, 3, 4
29. I. Leontiadis and M. Li. Secure and collusion-resistant data aggregation from convertible tags. *International Journal of Information Security*, 20(1):1–20, 2021. 3, 5

30. Q. Li and G. Cao. Efficient privacy-preserving stream aggregation in mobile sensing with low aggregation error. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 60–81. Springer, 2013. 3, 4
31. H. Liu, T. Gu, Y. Liu, J. Song, and Z. Zeng. Fault-tolerant privacy-preserving data aggregation for smart grid. *Wireless Communications and Mobile Computing*, 2020:1–10, 2020. 4
32. P. Longa and M. Naehrig. Speeding up the number theoretic transform for faster ideal lattice-based cryptography. In *Cryptology and Network Security: 15th International Conference, CANS 2016, Milan, Italy, November 14-16, 2016, Proceedings 15*, pages 124–139. Springer, 2016. 7
33. V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. *Journal of the ACM (JACM)*, 60(6):1–35, 2013. 4
34. F. D. McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 19–30, 2009. 13
35. R. Rabaninejad, A. Bakas, E. Frimpong, and A. Michalas. A secure bandwidth-efficient treatment for dropout-resistant time-series data aggregation. In *2023 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, pages 640–645. IEEE, 2023. 3, 5
36. A. Roy Chowdhury, C. Wang, X. He, A. Machanavajjhala, and S. Jha. Cryptε: Crypto-assisted differential privacy on untrusted servers. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 603–619, 2020. 6
37. E. Shi, H. Chan, E. Rieffel, R. Chow, and D. Song. Privacy-preserving aggregation of time-series data. In *Annual Network & Distributed System Security Symposium (NDSS)*. Internet Society., 2011. 2, 3, 4, 5, 6, 8
38. J. Takeshita, Z. Carmichael, R. Karl, and T. Jung. TERSE: tiny encryptions and really speedy execution for post-quantum private stream aggregation. In *International Conference on Security and Privacy in Communication Systems*, pages 331–352. Springer, 2022. 4
39. J. Takeshita, R. Karl, T. Gong, and T. Jung. SLAP: simpler, improved private stream aggregation from ring learning with errors. *Journal of Cryptology*, 36(2):8, 2023. 2, 4, 5, 6, 11, 12, 15, 16
40. H. Tian, F. Zhang, Y. Shao, and B. Li. Secure linear aggregation using decentralized threshold additive homomorphic encryption for federated learning. *arXiv preprint arXiv:2111.10753*, 2021. 2, 3
41. S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, and Y. Zhou. A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM workshop on artificial intelligence and security*, pages 1–11, 2019. 2
42. F. Valovich and F. Alda. Computational differential privacy from lattice-based cryptography. In *International Conference on Number-Theoretic Methods in Cryptology*, pages 121–141. Springer, 2017. 6
43. H. Waldner, T. Marc, M. Stopar, and M. Abdalla. Private stream aggregation from labeled secret sharing schemes. *Cryptology ePrint Archive*, 2021. 11
44. Y. Wang, A. Zhang, S. Wu, and S. Yu. Vosa: Verifiable and oblivious secure aggregation for privacy-preserving federated learning. *IEEE Transactions on Dependable and Secure Computing*, 2022. 2
45. J. Won, C. Y. Ma, D. K. Yau, and N. S. Rao. Proactive fault-tolerant aggregation protocol for privacy-assured smart metering. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pages 2804–2812. IEEE, 2014. 2, 3, 4