Collusion-Tolerable Privacy-Preserving Sum and Product Calculation without Secure Channel

Taeho Jung[†], Xiang-Yang Li[†§], Meng Wan[‡]
[†]Department of Computer Science, Illinois Institute of Technology, Chicago, IL
[§]Department of Computer Science and Technology and TNLIST, Tsinghua University, Beijing
[‡]Center for Science and Technology Development, Ministry of Education, Beijing

Abstract—Much research has been conducted to securely outsource multiple parties' data aggregation to an untrusted aggregator without disclosing each individual's privately owned data, or to enable multiple parties to jointly aggregate their data while preserving privacy. However, those works either require secure pair-wise communication channels or suffer from high complexity. In this paper, we consider how an external aggregator or multiple parties can learn some algebraic statistics (e.g., sum, product) over participants' privately owned data while preserving the data privacy. We assume all channels are subject to eavesdropping attacks, and all the communications throughout the aggregation are open to others. We first propose several protocols that successfully guarantee data privacy under semi-honest model, and then present advanced protocols which tolerate up to k passive adversaries who do not try to tamper the computation. Under this weak assumption, we limit both the communication and computation complexity of each participant to a small constant. At the end, we present applications which solve several interesting problems via our protocols.

Keywords: Privacy, data aggregation, secure channels, SMC, homomorphic.

1 Introduction

The Privacy-preserving data aggregation problem has long been a hot research issue in the field of applied cryptography. In numerous real life applications such as crowd sourcing or mobile cloud computing, individuals need to provide their sensitive data (location-related or personal-information-related) to receive specific services from the system (e.g., location based services or mobile based social networking services). There are usually two different models in this problem: 1) an external aggregator collects the data and wants to conduct an aggregation function on participants' data (e.g., crowd sourcing); 2) participants themselves are willing to jointly compute a specific aggregation function whose input data is co-provided by themselves (e.g., social networking services). However, the individuals' data should be kept secret, and the aggregator or other participants are not supposed to learn any useful information about it. Secure Multi-party Computation (SMC), Homomorphic Encryption (HE) and other cryptographic methodologies can be partially or fully exploited to solve this problem, but they are subject to some restrictions in such problems.

Secure Multi-party Computation (SMC) was first formally introduced by Yao [39] in 1982 as Secure Two-Party Computation. Generally, it enables n parties who want to jointly and privately compute a function

$$f(x_1, x_2, \cdots, x_n) = \{y_1, y_2, \cdots, y_n\}$$

where x_i is the input of the participant i, and the result y_i is returned to the participant i only. Each result can be relevant to all input x_i's, and each participant i knows nothing but his own result y_i. One could let the function in SMC output only one uniform result to all or parts of participants, which is the algebraic aggregation of their input data. Then the privacy-preserving data aggregation problem seems to be solved by SMC. However this actually does not completely solve it because interactive invocation is required for participants in synchronous SMC (e.g., [21]) , which leads to high communication and computation complexity (compared in the Section 7). Even in the asynchronous SMC, the computation complexity is still too high for practical applications.

Homomorphic Encryption (HE) allows direct addition and multiplication of ciphertexts while preserving decryptability. That is, $\text{Enc}(m_1) \otimes \text{Enc}(m_2) = \text{Enc}(m_1 \times m_2)$, where $\text{Enc}(m)$ stands for the ciphertext of m, and $\otimes$, $\times$ refer to the homomorphic operations on the ciphertext and plaintexts respectively. One could also try to solve our problem using this technique, but HE uses the same decryption key for original data and the aggregated data. This forbids aggregator from decrypting the aggregated result, because if the aggregator were allowed to decrypt the final result, he could also decrypt the

*Dr. Xiang-Yang Li is the contact author of this paper.

1. The research of Li is partially supported by NSF CNS-1035894, NSF ECCS-1247944, NSF ECCS-1343306, National Natural Science Foundation of China under Grant No. 61170216, No. 61228202. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of author(s) and do not necessarily reflect the views of the funding agencies (NSF, and NSFC).

Secure Multi-party Computation	
Pros	different outputs for different participants
Cons	high complexity due to the computation based on garbled circuit frequent interactions required for synchronous SMC
Homomorphic Encryption	
Pros	efficient if # of multiplcations is restricted
Cons	decrypter can decrypt both aggregated data and individual data trade-off between # of multiplications and complexity exists

individual ciphertext, which contradicts our motivation. Also, because the size of the plaintext space is limited, the number of addition and multiplication operations executed on ciphertexts was limited until Gentry *et al.* proposed a fully homomorphic encryption scheme [19] and implemented it in [20]. However, Naehrig *et al.* pointed out in [34] that the complexity of such fully homomorphic encryption is too high to use in real applications. Naehrig *et al.* also proposed a HE scheme which sacrificed the number of multiplications for speed, but it still needs too much time to execute homomorphic operations on ciphertexts (also compared in Section 7).

Besides the aforementioned drawbacks, both SMC and HE require an initialization phase during which participants request keys from key issuers via secure channel. This could be a security bottleneck since the security of those schemes relies on the assumption that keys are delivered to authorized participants only.

In this paper, we revisit the classic privacy preserving data aggregation problem. Our goal is to design efficient protocols without relying on a trusted authority or secure pair-wise communication channels. The main contributions of this paper are:

1) *Formulation of a model without secure channel or trusted center*: Different from many other models in privacy-preserving data aggregation problem, our model does not require a secure communication channel nor a trusted central key issuer.
2) *Efficient protocol in linear time*: The total communication and computation complexity of our work is proportional to the number of participants n, while the complexities of many similar works are proportional to n^2. We do not use complicated encryption protocols, which makes our system much faster than other proposed systems.
3) *Secure sum and product calculation*: We generalize the privacy-preserving data aggregation to multivariate sum and product calculation whose inputs are jointly provided by multiple parties. That is, our scheme enables multiple parties to securely compute
$$\sum_i x_i \quad \text{or} \quad \prod_i x_i$$
where x_i is a privately known data by user i.
4) *Tolerate up to k collusive adversaries*: Our protocol is robust against up to k colluding passive adversaries who do not try to tamper the computation.

The rest of the paper is organized as follows. We present the system model and necessary background in Section 3. In Section 4, we analyze the needed number of communications with secure communication channels when users communicate randomly. We address the privacy preserving sum and product calculation in Section 5 by presenting two types of efficient protocols. Then, we present detailed security analysis of our protocols in Section 6, and the performance evaluation is reported in Section 7. Finally, solutions to various problems based on our protocols are presented in Section 8 and finally conclude the paper with the discussion of some future work in Section 9.

2 Related Work

Many novel protocols have been proposed for privacy-preserving data aggregation or in general secure multi-party computation. Castelluccia *et al.* [5] presented a provable secure and efficient aggregation of encrypted data in WSN, which is extended from [6]. They designed a symmetric key homomorphic encryption scheme which is additively homomorphic to conduct the aggregation operations on the ciphertexts. Their scheme uses modular addition, so the scheme is good for CPU-bounded devices such as sensor nodes in WSN. Their scheme can also efficiently compute various statistical values such as mean, variance and deviation. However, since they used the symmetric homomorphic encryption, their aggregator can decrypt each individual sensor's data, and they assumed the trusted aggregator in their model.

Sheikh *et al.* [36] proposed a k-secure sum protocol, which is motivated by the work of Clifton *et al.* [9]. They significantly reduced the probability of data leakage in [9] by segmenting the data block of individual party, and distributing segments to other parties. Here, sum of each party's segments is his data, therefore the final sum of all segments are sum of all parties' data. This scheme can be easily converted to k-secure product protocol by converting each addition to multiplication. However, pair-wise unique secure communication channels should be given between each pair of users such that only the receiver and the sender know the transmitted segment. Otherwise, each party's secret data can be calculated by performing $O(k)$ computations. In this paper, we remove the limitation of using secure communication channels.

The work of He *et al.* [23] is similar to Sheikh *et al.*'s work. They proposed two privacy-preserving data aggregation schemes for wireless sensor networks: the Cluster-Based Private Data Aggregation (CPDA) and the Slice-Mix-AggRegaTe (SMART). In CPDA, sensor nodes form clusters randomly and collectively compute the aggregate result within each cluster. In the improved SMART, each node segments its data into n slices and distributes $n-1$ slices to nearest nodes via secure channel. However, they only supports additions, and since each data is segmented, communication overhead per node is linear to the number of slices n.

Shi *et al.* [37] proposed a construction that n participants periodically upload encrypted values to an aggregator, and the aggregator computes the sum of those

values without learning anything else. This scheme is close to our solution, but they assumed a trusted key dealer in their model. The key dealer distributes random key k_i to participant i and key k_0 to the aggregator, where $\Pi_{i=0}^{n} k_i = 1$, and the ciphertext is in the format of $C_i = k_i \cdot g^{x_i}$. Here, g is a generator, k_i is a participant's key and x_i is his data (for $i = 1, 2, \cdots n$). Then, the aggregator can recover the sum $\sum_{i=1}^{n} x_i$ iff he received ciphertexts from all of the participants. He computes $k_0 \Pi_{i=1}^{n} C_i$ to get $g^{\sum_{i=1}^{n} x_i}$, and uses brute-force search to find the $\sum_{i=1}^{n} x_i$ or uses Pollard's lambda method [33] to calculate it. This kind of brute-force decryption limits the space of plaintext due to the hardness of the discrete logarithm problem, otherwise no deterministic algorithm can decrypt their ciphertext in polynomial time. The security of their scheme relies on the security of keys k_i. Later, Joye *et al.* [25] proposed binomial property based solution to efficiently decrypt the sum, but they also rely on a trusted key dealer who distributes the keys.

In our scheme, the trusted aggregator in [5][6] is removed since data privacy against the aggregator is also a top concern these days. Unlike [23][36], we assumed insecure channels, which enabled us to get rid of expensive and vulnerable key pre-distribution. We did not segment each individual's data, our protocols only incur constant communication overhead for each participant. Our scheme is also based on the hardness of the discrete logarithm problem like [37], but we do not trivially employ brute-force manner in decryption, instead, we employ our novel efficient protocols for sum and product calculation.

Besides, there are also several works ([7], [24], [22], [31]) which have similar goals as ours. They are presented to achieve collusion-resistant or fault-tolerable data aggregation with privacy preservation. However, they leverage the differential privacy [10], [43], [17], [11] in various ways to achieve privacy as well as collusion (or fault) tolerance. The results given by their solutions contain asymptotically bounded error, and those works may not be applicable when exact results are desired.

3 System Model

3.1 Problem Definition and Threat Model

Assume that there are n participants $\{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_n\}$, and each participant $\mathbf{p}_i$ has a privately known data x_i from $\mathbb{Z}_p$. The privacy-preserving data aggregation problem is to compute sum or product of x_i jointly or by an aggregator while preserving the data privacy. That is, the objective of the aggregator or the participants is to compute the following polynomial without knowing any individual x_i:

$$f(\mathbf{x}) = \sum_{i=1}^{n} x_i \quad \text{or} \quad f(\mathbf{x}) = \prod_{i=1}^{n} x_i \tag{1}$$

Here vector $\mathbf{x} = (x_1, x_2, \cdots, x_n)$. For simplicity, we assume that the final result $f(\mathbf{x})$ is positive and bounded from above by a large prime number P.

We employ two different models in this paper: *One Aggregator Model* and *Participants Only Model*. These two models are general cases we are faced with in real applications.

One Aggregator Model: In the first model, we have one aggregator $\mathcal{A}$ who wants to compute the function $f(\mathbf{x})$. We assume the aggregator is **untrustful** and **curious**. That is, he always eavesdrops the communications between participants and tries to harvest their input data, but he will follow the protocol specification. We also assume participants do not trust each other and that they are curious as well, *i.e.*, they also eavesdrop all the communications and follow the protocol specification. We could also consider having multiple aggregators, but this is a simple extension which can be trivially achieved from our first model. We call this model the *One Aggregator Model*. Note that in this model, any single participant $\mathbf{p}_i$ is not allowed to compute the final result $f(\mathbf{x})$.

Participants Only Model: The second model is similar to the first one except that there are n participants only and there is no aggregator. In this model, all the participants are equal and they all will calculate the final aggregation result $f(\mathbf{x})$.

We further assume the participants and aggregator may be **passively adversarial**. That is, they will not tamper the computation, but they may try to manipulate their calculation to infer others' private values.

3.2 Security Model

Firstly, we assume that all the communication channels in our protocol are insecure. Anyone can eavesdrop them to intercept the data being transferred. To address the challenges of insecure communication channel, we assume that the following CDH problem is computationally intractable, *i.e.,* any probabilistic polynomial time adversary has negligible chance to solve the following problem:

Definition 1 (CDH Problem in $\mathbb{G}$)**.** *The Computational Diffie-Hellman problem in a multiplicative group $\mathbb{G}$ with generator g is defined as follows: given only $g, g^a, g^b \in \mathbb{G}$ where $a, b \in \mathbb{Z}$, compute g^{ab} without knowing a or b.*

Additionally, similar Decisional Deffie-Hellman (DDH) problem is defined as follows:

Definition 2 (DDH Problem in $\mathbb{G}$)**.** *The Decisional Diffie-Hellman problem in a multiplicative group $\mathbb{G}$ with generator g is defined as follows: given only $g, g^a, g^b, g^c \in \mathbb{G}$ where $a, b, c \in \mathbb{Z}$, decide if $g^{ab} = g^c$.*

Obviously, anyone who solves the CDH problem can solve the DDH problem by computing g^{ab} based on g^a, g^b, but not vice versa. That is, the DDH problem is easier than the CDH one, and if a problem is as hard as a DDH problem, it is harder than a CDH problem. Our protocol is based on the assumption that it is computational expensive to solve the CDH problem as in other

similar research works ([41], [28], [26], [27], [32], [42]). Then, we define the security of our privacy-preserving sum and product calculation as follows.

Definition 3 (CDH-Security in $\mathbb{G}$)**.** *We say our privacy-preserving (sum or product) calculation is CDH-secure in $\mathbb{G}$ if any Probabilistic Polynomial Time Adversary (PPTA) who cannot solve the CDH problem with non-negligible chance has negligible chance to infer any honest participant's private value in $\mathbb{G}$, i.e., any PPTA's probability to solve the CDH problem ϵ satisfies $\epsilon < |\frac{1}{p(\kappa)}|$ for any polynomial $p(\cdot)$ where κ is the order of the group $\mathbb{G}$ defined in the CDH problem.*

Informally, we say our calculation is CDH-secure in $\mathbb{G}$ if illegally inferring an honest participant's private value during our calculation is at least as hard as CDH problem in $\mathbb{G}$.

4 Achieving Sum & Product Under Secure Channel

Before introducing our aggregation scheme without secure communication channel, we first describe the basic idea of randomized secure sum calculation under secured communication channel (It can be trivially converted to secure product calculation). The basic idea came from Clifton *et al.* [9], which is also reviewed in [38], but we found their setting imposed unnecessary communication overhead, and we reduced it while maintaining the same security level. Assume participants $\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_n$ are arranged in a ring for computation purpose. Each participant $\mathbf{p}_i$ itself breaks its privately owned data block x_i into k segments $s_{i,j}$ such that the sum of all k segments is equal to the value of the data block. The value of each segment is randomly decided. For sum, we can simply assign random values to segments $s_{i,j}$ $(1 \leq j \leq k-1)$ and let $s_{i,k} = x_i - \sum_{j=1}^{k-1} s_{i,j}$. Similar method can be used for product. In this scheme, each participant randomly selects $k-1$ participants and transmit each of those participants a distinctive segment $s_{i,j}$. Thus at the end of this redistribution each of participants holds several segments within which one segment belongs to itself and the rest belongs to some other participants. The receiving participant adds all its received segments and transmits its result to the next participant in the ring. This process is repeated until all the segments of all the participants are added and the sum is announced by the aggregator.

Recall that there are n participants and each participant randomly selects $k-1$ participants to distribute its segments. Clearly, a larger k provides better computation privacy, however it also causes larger communication overhead which is not desirable. In the rest of this section, we are interested at finding an appropriate k in order to reduce the communication cost while preserving computation privacy.

In particular, we aim at selecting the smallest k to ensure that each participant holds at least one segment from the other participants after redistribution. We can view this problem as placing identical and indistinguishable balls into n distinguishable (numbered) bins. This problem has been extensively studied and well-understood and the following lemma can be proved by simple union bound:

Lemma 4.1. *Let $\epsilon \in (0,1)$ be a constant. If we randomly place $(1+\epsilon)n\ln n$ balls into n bins, with probability at least $1-\frac{1}{n^\epsilon}$, all the n bins are filled.*

Assume that each participant will *randomly* select $k-1$ participants (including itself) for redistribution. By treating each round of redistribution as one trial in coupon's collector problem, we are able to prove that each participant only needs to redistribute $((1+\epsilon)n\ln n)/n = (1+\epsilon)\ln n$ segments to other participants to ensure that every participant receives at least one segment with high probability. However, different from previous assumption, each participant will select $k-1$ participants except itself to redistribute its segments in our scheme. Therefore, we need one more round redistribution for each participant to ensure that every participant will receive at least one copy from other participants with high probability.

Theorem 4.2. *Let $\epsilon \in (0,1)$ be a constant. If each participant randomly selects $(1+\epsilon)\ln n + 1$ participants to redistribute its segments, with probability at least $1-\frac{1}{n^\epsilon}$, each participant receives at least one segment from the other participants.*

This theorem reveals that by setting k to the order of $\ln n$, we are able to preserve the computation privacy. Compared with traditional secure sum protocol, our scheme dramatically reduce the communication complexity. However, we assume that the communication channel among participants are secure in above scheme. In the rest of this paper, we try to tackle the secure aggregation problem under unsecured channels.

5 Sum and Product Calculation without Secure Channel

In this section, we first present two novel calculation protocols for each model which calculate sum and product while preserving each participant's data privacy against semi-honest adversaries who follow the protocol specification and do not collude with anyone. Then, we show how to defend against passive adversaries who may adaptively choose their secret parameters based on others' public parameters and collude with each other.

5.1 Definitions

Table 1 summarizes the notations frequently used in this paper.

The groups $\mathbb{G}_1, \mathbb{G}_2$ are selected as follows. Two large same-length prime numbers p, q are chosen such that q divides $p-1$. Then, the q-order cyclic multiplicative

TABLE 1
Frequently used notations

$\mathbf{p}_i$	i-th participant in data aggregation
$\mathcal{A}$	Aggregator
$\mathbb{Z}_q$	The ring $\mathbb{Z}/q\mathbb{Z}$
$\mathbb{G}_1, g_1$	Multiplicative group for product protocol, and its generator
$\mathbb{G}_2, g_2$	Multiplicative group for sum protocol, and its generator

group $\mathbb{G}_1$ is defined as $\langle g_1 \rangle$ where the generator g_1, with a random number $h \in \mathbb{Z}_p$, is selected as:

$$g_1 = h^{(p-1)/q} \mod p \quad s.t. \quad g_1 \neq 1 \mod p$$

Similarly, the q-order multiplicative group $\mathbb{G}_2$ is defined as $\langle g_2 \rangle$ where the generator $g_2 = g_1^p \mod p^2$.

5.2 Product Protocol - Participants Only Model

In this model, the participants want to jointly compute the value $f(\mathbf{x}) = \prod_i x_i$ with their privately known values $x_i \in \mathbb{Z}_p$, where p is a large prime. The basic idea of our protocol is to find some random integers $R_i \in \mathbb{G}_1$ such that $\prod_i R_i = 1 \mod p$ and the user $\mathbf{p}_i$ can compute the random number R_i easily while it is computationally expensive for other participants to compute the value R_i.

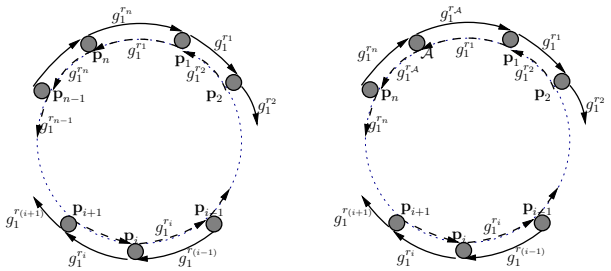
Our protocol for privacy preserving product calculation $\prod_i x_i$ is composed of the following three algorithms: Setup, Encrypt, Product.

Setup $\rightarrow r_i \in \mathbb{Z}_q, R_i = (g_1^{r_{i+1}}/g_1^{r_{i-1}})^{r_i} \in \mathbb{G}_1$

We assume all participants are arranged in a circle for computation purpose. The circle can be formed according to any order (*e.g.*, lexicographical order of the MAC address or the geographical location). This is irrelevant to the security of our calculation and omit further discussions about it.

Every $\mathbf{p}_i (i \in \{1, \cdots, n\})$ randomly chooses a secret integer $r_i \in \mathbb{Z}_q$, and calculates a public parameter $g_1^{r_i} \in \mathbb{G}_1$. Then, each $\mathbf{p}_i$ shares $Y_i = g_1^{r_i} \in \mathbb{G}_1$ with $\mathbf{p}_{i-1}$ and $\mathbf{p}_{i+1}$ (where $\mathbf{p}_{n+1} = \mathbf{p}_1$ and $\mathbf{p}_0 = \mathbf{p}_n$).

After a round of exchanges, the participant $\mathbf{p}_i$ computes the number $R_i = (Y_{i+1}/Y_{i-1})^{r_i} = (g_1^{r_{i+1}}/g_1^{r_{i-1}})^{r_i} \in \mathbb{G}_1$, and keeps the number R_i as a secret randomizer. Note that $\mathbf{p}_1$ calculates $(g_1^{r_2}/g_1^{r_n})^{r_1}$ and $\mathbf{p}_n$ calculates $(g_1^{r_1}/g_1^{r_{(n-1)}})^{r_n}$.



(a)Participants Only (b) One Aggregator

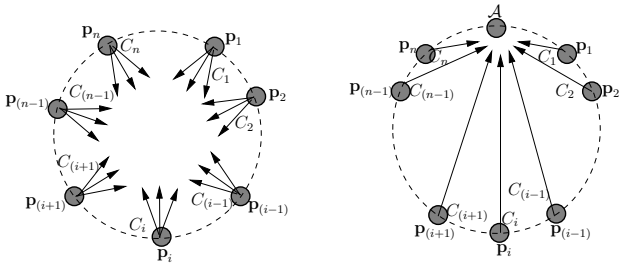
Fig. 1. Communications in Setup

Encrypt$(x_i) \rightarrow C_i \in \mathbb{Z}_p$

Every $\mathbf{p}_i$ creates the ciphertext:

$$C_i = x_i \cdot R_i = x_i \cdot (g_1^{r_{i+1}}/g_1^{r_{i-1}})^{r_i} \mod p$$

where $x_i \in \mathbb{Z}_p$ is his private input data. Then, he broadcasts this ciphertext.



(a)Participants Only (b) One Aggregator

Fig. 2. Communications in Encrypt

Product$(\{C_1, C_2, \cdots, C_n\}) \rightarrow \prod_{i=1}^n x_i \in \mathbb{Z}_p$

Any $\mathbf{p}_i$, after receiving $n-1$ ciphertexts from other $\mathbf{p}_i$'s, calculates the following product:

$$\begin{aligned}
\prod_{i=1}^n C_i &= \prod_{i=1}^n (x_i (g_1^{r_{i+1}}/g_1^{r_{i-1}})^{r_i}) \mod p \\
&= (\prod_{i=1}^n x_i) \prod_{i=1}^n ((g_1^{r_{i+1}}/g_1^{r_{i-1}})^{r_i}) \mod p \\
&= (\prod_{i=1}^n x_i) g_1^{\sum_{i=1}^n (r_{i+1} r_i - r_i r_{i-1})} \mod p \\
&= \prod_{i=1}^n x_i \mod p
\end{aligned}$$

where $r_{n+1} = r_1, r_0 = r_n$. To make sure that we can get a correct result $\prod_{i=1}^n x_i$ without modular, we need to choose p to be large enough, say $p \geq M^n$, where M is a known upper bound on x_i.

5.3 Product Protocol - One Aggregator Model

The product calculation in the One Aggregater Model is similar to the protocol above, except that the aggregator $\mathcal{A}$ acts as the $(n+1)$-th participant $\mathbf{p}_{n+1}$. The second difference is that, each participant $\mathbf{p}_i$ will send the ciphertext C_i to the aggregator, instead of broadcasting to all participants. However, since our communication channel is insecure, this is essentially the broadcast from the adversary's perspective. The last difference is that the aggregator $\mathcal{A}$ will *not* announce its random number $R_{n+1} = (g_1^{r_1}/g_1^{r_n})^{r_{n+1}}$ to any other participant.

Each participant $\mathbf{p}_{i \in [1,n]}$ sends the ciphertext $C_i = R_i \cdot x_i$ to the aggregator $\mathcal{A}$. The aggregator $\mathcal{A}$ then calculates

$$R_{n+1} \prod_{i=1}^n C_i = \prod_{i=1}^n x_i \mod p$$

to achieve the final product, where R_{n+1} is kept secret to any participant.

5.4 Sum Protocol - Participants Only Model

Now, we present how to let n participants jointly compute $f(\mathbf{x}) = \sum_{i=1}^{n} x_i$ given their privately known values $x_i \in \mathbb{G}_1$. It seems we can exploit the aforementioned method in product calculation by finding random numbers R_i such that $\sum_{i=1}^{n} R_i = 0 \mod p$, but we found it is challenging to let participants independently achieve such numbers. The previous method was secure because of the hardness of the discrete logarithm, but the same analogue does not exist in the additions.

The basic idea of this protocol is to convert the sum of numbers into product of numbers. Previous solution [37] essentially applied this approach also by computing the product of $\prod_{i=1}^{n} g^{x_i} = g^{\sum_{i=1}^{n} x_i}$ and finding $\sum_{i=1}^{n} x_i$ by computing the discrete logarithm of the product. As discrete logarithm is computationally expensive, we will not adopt this method. Instead, we propose a computationally efficient method here.

In a nutshell, we exploit the modular property below to achieve the privacy preserving sum protocol.

$$(1+p)^m = \sum_{i=0}^{m} \binom{m}{i} p^i = 1 + mp \mod p^2 \qquad (2)$$

From the Equation (2), we conclude that

$$\prod_{i=1}^{n} (1+p)^{x_i} = \prod_{i=1}^{n} (1 + p \cdot x_i) = (1 + p \sum_{i} x_i) \mod p^2.$$

Our protocol for privacy preserving sum calculation $\sum_i x_i$ has the following steps: Setup, Encrypt, Sum.

Setup $\rightarrow r_i \in \mathbb{Z}_q, R_i = (g_2^{r_{i+1}} / g_2^{r_{i-1}})^{r_i} \in \mathbb{G}_2$

Every $\mathbf{p}_i$ randomly picks a secret number $r_i \in \mathbb{Z}_q$, and calculates a public parameter $g_2^{r_i} \mod p^2$. Then, he shares $Y_i = g_2^{r_i} \in \mathbb{G}_2$ with $\mathbf{p}_{i+1}$ and $\mathbf{p}_{i-1}$. Similar to the product calculation protocol, $\mathbf{p}_n$ shares his public parameter with his $\mathbf{p}_{(n-1)}$ and $\mathbf{p}_1$, and $\mathbf{p}_1$ shares his public parameter with $\mathbf{p}_2$ and $\mathbf{p}_n$.

After a round of exchanges, each $\mathbf{p}_i$ calculates $R_i = (g_2^{r_{i+1}} / g_2^{r_{i-1}})^{r_i} \mod p^2$ and keeps this as a secret randomizer.

Encrypt$(x_i, R_i) \rightarrow C_i \in \mathbb{Z}_{p^2}$

Every $\mathbf{p}_i$ first calculates $(1 + x_i \cdot p)$. Then, he multiplies the secret parameter $R_i = (g_2^{r_{i+1}} / g_2^{r_{i-1}})^{r_i}$ to it to get the ciphertext:

$$C_i = (1 + x_i \cdot p) \cdot R_i \mod p^2$$

After all, each participant broadcasts his ciphertext to each other.

Sum$(\{C_1, C_2, \cdots, C_n\}) \rightarrow \sum_{k=1}^{n} x_i \in \mathbb{Z}_p$.

Each participant $\mathbf{p}_i$, after receiving the ciphertexts from all of other participants, calculates the following $C \in \mathbb{Z}_{p^2}$:

$$\begin{aligned} C &= \prod_{i=1}^{n} C_i \mod p^2 \\ &= \prod_{i=1}^{n} (1 + x_i p)(g_2^{r_{i+1}} / g_2^{r_{i-1}})^{r_i} \mod p^2 \\ &= (1 + p \sum_{i=1}^{n} x_i) g_2^{\sum_{i=1}^{n} r_{i+1} r_i - r_i r_{i-1}} \mod p^2 \\ &= (1 + p \sum_{i=1}^{n} x_i) \mod p^2 \end{aligned}$$

Then, he calculates $(C-1)/p = \sum_{i=1}^{n} x_i \mod p$ to recover the final sum, where the division is not the $(C-1)$ times $p^{-1} \mod p$ but the quotient of $(C-1)/p$.

5.5 Sum Protocol - One Aggregator Model

Similar to the product protocol for One Aggregator Model, everything is same except that $\mathcal{A}$ acts as $(n+1)$-th participant in this model. The participants send their ciphertexts to $\mathcal{A}$, and $\mathcal{A}$ calculates

$$C = R_{n+1} \prod_{i=1}^{n} C_i = (1 + p \sum_{i=1}^{n} x_i) \mod p^2$$

Then, he can compute the final sum result $\sum_{i=1}^{n} x_i$.

5.6 Advanced Protocols – Defence Against k Adversarial Participants

Our protocol is based on the circle-wise exchanges among the participants arranged in a circle, in which $\mathbf{p}_i$ exchanges his secret with $\mathbf{p}_{i-1}$ and $\mathbf{p}_{i+1}$ (equivalent to a broadcast due to the insecure channel). Since $\mathbf{p}_i$'s randomizer R_i is $g^{(r_{i+1} - r_{i-1}) r_i}$, this construction is vulnerable to well known passive rushing adversaries, who have access to the message sent from reliable players before they choose their messages ([15]).

During the circle-wise exchanges, an adversarial participant $\mathbf{p}_i$ may send out $g^{r_{i+2} - a}$ to $\mathbf{p}_{i+1}$ $(a \in \mathbb{Z})$ after receiving $g^{r_{i+2}}$ from $\mathbf{p}_{i+2}$. Then, $\mathbf{p}_{i+1}$'s randomizer is equal to $g^{(r_{i+2} - (r_{i+2} - a)) r_{i+1}} = g^{a r_{i+1}}$, which can be efficiently solved with $\mathbf{p}_{i+1}$'s public parameter $g^{r_{i+1}}$. This attack does not tamper the final computation and cannot be detected.

In a word, the rushing adversary can manipulate the value $r_{i+1} - r_{i-1}$. Therefore, we present an advanced product protocol which is secure against such attackers. For simplicity, we only show the advanced construction of product calculation protocol in Participants Only model, but the analogues in the other settings are easy to derive.

Setup $\rightarrow r_i \in \mathbb{Z}_q, R_i = \left(g_1^{r_{i+2}} / g_1^{r_{i-1}}\right)^{r_{i+1} r_i}$.

Every $\mathbf{p}_i$ randomly picks a secret number $r_i \in \mathbb{Z}_q$, and calculates a public parameter $g_1^{r_i} \in \mathbb{G}_1$. Then, each $\mathbf{p}_i$

shares $Y_i = g_1^{r_i} \in \mathbb{G}_1$ with $\mathbf{p}_{i-1}, \mathbf{p}_{i+1}$ and $\mathbf{p}_{i+2}$ (where $\mathbf{p}_{n+1} = \mathbf{p}_1$, $\mathbf{p}_0 = \mathbf{p}_n$ and $\mathbf{p}_{n+2} = \mathbf{p}_2$).

After a round of exchanges, the participant $\mathbf{p}_{i+1}$ computes the number

$$Y'_{i+1} = (Y_{i+2}/Y_{i-1})^{r_{i+1}} = \left(g_1^{r_{i+2}}/g_1^{r_{i-1}}\right)^{r_{i+1}}$$

and sends Y'_{i+1} to $\mathbf{p}_i$. After the second round of exchanges, the participant $\mathbf{p}_i$ computes

$$R_i = Y'^{r_i}_{i+1} = \left(g_1^{r_{i+2}}/g_1^{r_{i-1}}\right)^{r_{i+1}r_i}$$

and keeps it as a secret randomizer.

Encrypt$(x_i, R_i) \to C_i \in \mathbb{Z}_p$

Every $\mathbf{p}_i$ calculates

$$C_i = x_i \cdot R_i = x_i \left(g_1^{r_{i+2}}/g_1^{r_{i-1}}\right)^{r_{i+1}r_i} \mod p$$

Then, he broadcasts C_i as his ciphertext.

Product$(\{C_1, C_2, \cdots, C_n\}) \to \prod_{i=1}^n x_i \in \mathbb{Z}_p$

Any $\mathbf{p}_i$, after receiving $n-1$ ciphertexts from other participants, calculates the following product:

$$\begin{aligned}
\prod_{i=1}^n C_i &= \prod_{i=1}^n \left(x_i \left(g_1^{r_{i+2}}/g_1^{r_{i-1}}\right)^{r_{i+1}r_i}\right) \mod p \\
&= \left(\prod_{i=1}^n x_i\right) \left(\prod_{i=1}^n \left(g_1^{r_{i+1}}/g_1^{r_{i-1}}\right)^{r_{i+1}r_i}\right) \mod p \\
&= \left(\prod_{i=1}^n x_i\right) g_1^{\sum_{i=1}^n (r_{i+2}r_{i+1}r_i - r_{i+1}r_ir_{i-1})} \mod p \\
&= \prod_{i=1}^n x_i \mod p
\end{aligned}$$

where $r_{n+1} = r_1$, $r_0 = r_n$ and $r_{n+2} = r_2$.

With this construction, a passive adversary $\mathbf{p}_{i-1}$ cannot compute R_i even if he launches a passive rushing attack and get the value of $r_{i+2} - r_{i-1}$ because r_{i+1} remains unknown to him. If $\mathbf{p}_{i-1}$ colludes with $\mathbf{p}_{i+1}$, we can add another round of exchanges such that $\mathbf{p}_i$'s randomizer R_i is equal to

$$R_i = \left(g_1^{r_{i+3}}/g_1^{r_{i-1}}\right)^{r_{i+2}r_{i+1}r_i} \mod p$$

Then, two colluding rushing adversaries cannot calculate the randomizer. In general, when there are k colluding adversaries, we can have $k+1$ rounds of exchanges such that $\mathbf{p}_i$'s randomizer is equal to

$$R_i = \left(g_1^{r_{i+k+1}}/g_1^{r_{i-1}}\right)^{r_{i+k}r_{i+k-1}\cdots r_{i+2}r_{i+1}r_i} \mod p$$

5.7 Further Improvement

In fact, the Setup algorithm must be repeated every time a new sum or product is desired, in which two communications per participant are required. Otherwise, the random number R_i remains same, which leaks much information. Each participant needs to communicate with two other participants in every calculation, but we can remove this process in One Aggregator Model by leveraging the Joye *et al*'s work [25].

Briefly, Joye *et al* let a trusted key dealer distribute a set of random numbers s_i's to the participants and $s_0 = -\sum s_i$ to the aggregator. Then, it satisfies that $\prod H(t)^{s_i} = 1$, where t is the timestamp and $H(\cdot)$ is any hash function, and this property can be used to conduct sum or product calculation with privacy preservation, where the encryption is very similar to the one in this paper. Since $H(t)$ changes along the time, participants get fresh random numbers without communication. Therefore, we can achieve a more communication efficient sum calculation protocol, and also remove the necessity of a trusted key dealer in [25] as follows.

Firstly, every participant picks a random number $s_i \in \mathbb{Z}_p$. Then, all participants and the aggregator engage in our privacy-preserving sum calculation (Section 5.5) to let the aggregator calculate $\sum s_i$. After the aggregator sets $s_0 = -\sum s_i$, he can compute $\sum x_i$ without knowing individual x_i using the data aggregation scheme in [25].

By doing so, participants do not need to repeat the exchanges to get new random numbers, and this allows them to get rid of the communications with other participants. This may be greatly reduce the communication overhead when the aggregator is a large server with whom the communication is cheap, but the P2P communications with other participants are expensive. Since the participants and the aggregator have $s_0, \cdots, s_n$ such that $\prod H(t)^{s_i} = 1$, the privacy-preserving product calculation can also be achieved as in Section 5.3.

6 Security Analysis

6.1 Restriction of the Product and Sum Protocol

In both protocols, we require that the number of participants is at least 3 in Participants Only Model and at least 2 in One Aggregator Model. In Participants Only Model, if there are only 2 participants, privacy is not preservable since it is impossible to let $\mathbf{p}_1$ know x_1+x_2 or x_1x_2 without knowing x_2. However, in One Aggregator Model, since only the aggregator $\mathcal{A}$ knows the final result, as long as there are two participants, $\mathcal{A}$ is not able to infer any individual's input data.

6.2 Security of Product Protocol

Since our communication channel is insecure, any adversary has same view in both Participants Only Model and One Aggregator Model. Thus, we only analyze the security of Participants Only Model in this section.

Theorem 6.1. *Our product protocol in Participants Only Model is CDH-secure in* $\mathbb{G}_1$.

Proof: In a nutshell, we show that any PPTA who has significant chance to infer private values in our product protocol has non-negligible advantage to solve the CDH problem, which is a contradiction to our security assumption that CDH problem is intractable.

For any ciphertext in a product protocol, we have

$$C_i = x_i(g_1^{r_{i+1}}/g^{r_{i-1}})^{r_i} = x_i g_1^{(r_{i+1}-r_{i-1})r_i} \mod p$$

To infer x_i given $C_i = x_i R_i$, any adversary has to solve the secret randomizer $R_i = g_1^{(r_{i+1}-r_{i-1})r_i}$ to infer x_i correctly. Note that any PPTA is only given $Y_{i_1} = g_1^{r_{i-1}}, Y_i = g_1^{r_i}$ and $Y_{i+1} = g_1^{r_{i+1}}$ from the insecure communication channel, then a PPTA can have $Y_{i+1}/Y_{i-1} = g_1^{r_{i+1}-r_{i-1}}$ and $Y_i = g_1^{r_i}$. Assume there exists a PPTA who can solve $g_1^{(r_{i+1}-r_{i-1})r_i}$, then he also solves the CDH problem defined in our group $\mathbb{G}_1$ and the easier DDH problem too. However, even the easier DDH problem is believed to be intractable in a k-th residues modulo a prime number ([4]), therefore any PPTA has a negligible advantage to solve $g_1^{(r_{i+1}-r_{i-1})r_i}$ in $\mathbb{G}_1$.

That is, inferring private values during our product protocol in Participants Only Model is at least as hard as a CDH problem in $\mathbb{G}_1$ for any Probabilistic Polynomial Time Adversary (PPTA), and thus the protocol is CDH-secure in $\mathbb{G}_1$. □

6.3 Security of Sum Protocol

Again, we only analyze the security in Participants Model Only. Analogously, we have the following theorem.

Theorem 6.2. *Our sum protocol in Participants Only Model is CDH-secure in* $\mathbb{G}_2$.

Proof: Similar to the aforementioned proof, in order to infer x_i given $C_i = (1+x_i p)R_i$, any PPTA has to solve the secret randomizer $R_i = g_2^{r_{i+1}-r_{i-1}r_i}$ first. Note that any PPTA is only given $Y_{i_1} = g_2^{r_{i-1}}, Y_i = g_1^{r_i}$ and $Y_{i+1} = g_2^{r_{i+1}}$ from the insecure communication channel, then a PPTA can have $Y_{i+1}/Y_{i-1} = g_2^{r_{i+1}-r_{i-1}}$ and $Y_i = g_2^{r_i}$. Assume there exists a PPTA who can solve $g_1^{(r_{i+1}-r_{i-1})r_i}$, then he also solves the CDH problem defined in our group $\mathbb{G}_2$. However, the CDH problem is proved to be hard in the finite field $\mathbb{F}_{p^2} = \mathbb{Z}/p^2\mathbb{Z}$ ([14]), therefore any PPTA has a negligible advantage to solve $g_2^{(r_{i+1}-r_{i-1})r_i}$ in $\mathbb{G}_2$.

That is, inferring private values during our sum protocol in Participants Only Model is at least as hard as a CDH problem in $\mathbb{G}_2$ for any Probabilistic Polynomial Time Adversary (PPTA), and thus the protocol is CDH-secure in $\mathbb{G}_2$. □

6.4 Security of Advanced Protocols

For the sake of simplicity, we only analyze the security of the advanced product protocol tolerant to k adversaries in Participants Only Model, but the same-level security is achieved in other settings, and the similar proofs holds for every setting.

Theorem 6.3. *The advanced product protocol tolerant to* k *adversaries in Participants Only Model is CDH-secure in* $\mathbb{G}_1$.

Proof: In order to infer x_i given $C_i = x_i R_i$, any adversary has to solve the secret randomizer

$$R_i = \left(g_1^{r_{i+k+1}}/g_1^{r_{i-1}}\right)^{r_{i+k}r_{i+k-1}\cdots r_{i+2}r_{i+1}r_i} \mod p$$

In the worst case, a rushing adversary can manipulate the value $r_{i+k+1} - r_{i-1}$, and he can collude with $k-1$ adversarial participants to get $k-1$ random numbers r_i's in the exponent. However, there are $k+1$ random numbers r_i's in the exponent, and thus at least two random numbers remain unknown to the adversarial participants. Therefore, the exponent

$$(r_{i+k+1} - r_{i-1})r_{i+k}r_{i+k-1}\cdots r_{i+2}r_{i+1}$$

remains unknown. Therefore, the adversaries cannot solve R_i with $g_1^{r_i}$.

W.l.o.g., let $\mathbf{p}_{i-1}$ be the rushing adversary and let $\mathbf{p}_{i+1}, \cdots, \mathbf{p}_{i+k-1}$ be $k-1$ remaining adversarial participants. Then, the adversarial participants are given

$$\left(g_1^{r_{i+k+1}}/g_1^{r_{i-1}}\right)^{r_{i+k}r_{i+k-1}\cdots r_{i+2}r_{i+1}}$$

from the $k+1$-th round of the exchanges and the $g_1^{r_i}$ from the first round of the exchanges, and they have to solve R_i. However, this is exactly the CDH problem, which is assumed to be intractable.

That is, inferring private values during the advanced product protocol tolerant to k adversaries in Participants Only Model is at least as hard as a CDH problem in $\mathbb{G}_1$ for any Probabilistic Polynomial Time Adversary (PPTA), and thus the protocol is CDH-secure in $\mathbb{G}_1$. □

7 Performance Evaluation

7.1 Complexity

We discuss the computation and communication complexities of sum and product protocols in each model in this section.

7.1.1 One Aggregator Model

It is easy to see that the computation complexities of Setup, Encrypt and Product of the product protocol are $O(1)$, $O(1)$ and $O(n)$ respectively. Also, Encrypt is executed for m times by each participant and Product is executed for m times by the aggregator in the *Advanced Scheme*.

Every participant and the aggregator exchanges g^{r_i}'s with each adjacent neighbor in the ring, which incurs

communication of $O(|p|)$ bits in Setup, where $|p|$ represents the bit length of p. In Encrypt, each participant sends m ciphertexts $c_k \prod_{i=1}^{n} x_i^{d_{i,k}}$'s to the aggregator, so the communication overhead of Encrypt is $O(m|p|)$ bits. Since n participants are sending the ciphertexts to the aggregator, the aggregator's communication overhead is $O(mn|p|)$.

Similarly, the computation complexities of Setup, Encrypt and Sum in the sum protocol are $O(1)$, $O(1)$ and $O(m)$ respectively, and they are executed for only once in the scheme. Hence, the communication overhead of Setup, Encrypt and Sum are $O(|p^2|)$ bits, $O(|p^2|)$ bits and $O(m|p^2|)$ bits respectively ($|p^2|$ is the big length of p^2).

Note that $|p^2| = 2|p|$. Then, the total complexity of aggregator and participants are as follows:

TABLE 2
One Aggregator Model

Aggregator	Computation	Communication (bits)
Product (Product)	$O(mn)$	$O(mn\|p\|)$
Sum (sum)	$O(m)$	$O(m\|p\|)$
Per Participant	Computation	Communication (bits)
Setup (Product)	$O(1)$	$O(\|p\|)$
Encrypt (Product)	$O(m)$	$O(m\|p\|)$
Setup (sum)	$O(1)$	$O(\|p\|)$
Encrypt (sum)	$O(1)$	$O(\|p\|)$

7.1.2 Participants Only Model

In the Participants Only Model, participants broadcast ciphertexts to others, and calculates the products and sums themselves, therefore the complexities are shown as below:

TABLE 3
Participants Only Model

Per Participant	Computation	Communication (bits)
Setup (Product)	$O(1)$	$O(\|p\|)$
Encrypt (Product)	$O(m)$	$O(mn\|p\|)$
Product (Product)	$O(mn)$	$O(mn\|p\|)$
Setup (sum)	$O(1)$	$O(\|p\|)$
Encrypt (sum)	$O(1)$	$O(m\|p\|)$
Sum (sum)	$O(m)$	$O(m\|p\|)$

Note that the communication overhead is balanced in the Participants Only Model, but the system-wide communication overhead is increased a lot. In the One Aggregator Model, the system-wide communication overhead is:

$$O(mn|p|) + O(m|p|) + n \cdot O(|p|) = O(mn|p|) \quad \text{(bits)}$$

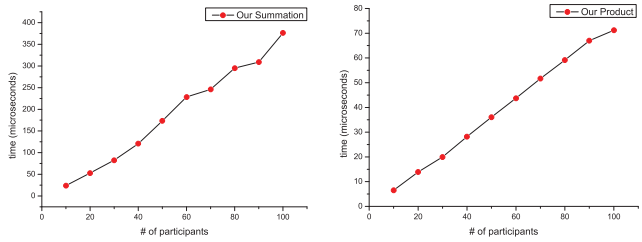
However, in the Participants Only Model, the system-wide communication complexity is:

$$n \cdot O(|p|) + n \cdot O(m|p|) + n \cdot O(mn|p|) = O(mn^2|p|) \quad \text{(bits)}$$

7.2 Evaluation by Implementation

We conduct extensive evaluations of our protocols. Our simulation result shows that the computation complexity of our protocol is indeed linear to the number of participants. To simulate and measure the computation overhead, we used GMP library to implement large number operations in our protocol in a computer with Intel i7-2620M @ 2.70GHz CPU and 2GB of RAM, and each result is the average time measured in the 100,000 times of executions. Also, the input data x_i is of 20-bit length, the q is of 256-bit length, and p is roughly of 270-bit length. That is, x_i is a number from $[0, 2^{20} - 1]$ and q is a uniform random number chosen from $[0, 2^{256} - 1]$.

In this simulation, we measured the total overhead of our novel product protocol and sum protocol (the second sum protocol) proposed in the Section 5). Here, we measured the total computation time spent in calculating the final result of n data (including encryption by n participants and the decryption by the aggregator). Since we only measure the computation overhead, there is no difference between One Aggregator Model and Participants Only Model.



(a) product (b) sum

Fig. 3. Running time for product and sum calculation.

First of all, the computation overhead of each protocol is indeed proportional to the number of participants. Also, the sum protocol needs much more time. This is natural because parameters in the sum protocol are in $\mathbb{Z}_{p^2}$, which are twice of the parameters in the product protocol in big length (they are in $\mathbb{Z}_p$).

Multivariate polynomial evaluation is composed of m products and one sum, so its computation overhead is barely the combination of the above two protocols' overhead.

We further compare the performance of our protocol with other existing multi party computation system implemented by Ben *et al.* [3] (FairplayMP). They implemented the BMR protocol [2], which requires constant number of communication rounds regardless of the function being computed. Their system provides a platform for general secure multi-party computation (SMC), where one can program their secure computation with Secure Function Definition Language (SFDL). The programs wrote in SFDL enable multiple parties to jointly evaluate an arbitrary sized boolean circuit. This boolean circuit is same as the garbled circuit proposed by Yao's 2 Party Computation (2PC) [39][40].

In Ben's setting, where they used a grid of computers, each with two Intel Xeon 3GHz CPU and 4GB of RAM, they achieved the computation time as following tables when they have 5 participants:

TABLE 4
Run time (milliseconds) of FairplayMP[3]

Gates	32	64	128	256	512	1024
Per Participant	64	130	234	440	770	1394

One addition of two k-bit numbers can be expressed with $k+1$ XOR gates and k AND gates. Therefore, if we set the length of input data as 20 bits (which is approximately 1 million), we need 41 gates per addition in FairplayMP system. When we conduct 26 additions (which is equivalent to 1066 gates) in our system, the total computation time is 72.2 microseconds, which is 2×10^4 times faster than the FairplayMP, which needs 1.394 seconds to evaluate a boolean circuit of 1024 gates. Even if we did not consider the aggregator's computation time in FairplayMP because they did not provide pure computation time (they provided the total run time including communication delay for the aggregator), our addition is already faster than their system. Obviously, the multiplication is much faster since it is roughly 8 times faster than the addition in our system.

We also compare our system with an efficient homomorphic encryption implementation [34]. Naehrig *et al.* proposed an efficient homomorphic encryption scheme which limits the total number of multiplications to a small number less than 100. If only one multiplication is allowed in their scheme (the fastest setting) and length of the modulus q is 1024, it takes 1 millisecond to conduct an addition and 41 milliseconds to conduct a multiplication. In our system, under the same condition, it takes 16.2 microseconds to conduct an addition and 0.7 microseconds to conduct a multiplication, which are approximately 100 times and 6×10^4 times faster respectively. They implemented the system in a computer with two Intel 2.1GHz CPU and 2GB of RAM. Even if considering our computer has a higher clock CPU, their scheme is still much slower than ours.

TABLE 5
Comparison between [34] and our system

	Addition	Multiplication
Lauter [34]	1 millisecond	41 milliseconds
Ours	16.2 microseconds	0.7 microseconds

The purpose of above two systems are quite different from ours, the first FairplayMP is for general multi-party computation and the second homomorphic encryption system is for general homomorphic encryption. They also provide a much higher level of security than ours since they achieve differential privacy, however, the comparison above does show the high speed of our system while our security level is still acceptable in real life applications, and this is one of the main contributions of this paper.

8 Applications

There are numerous interesting applications that can be achieved by our protocols. In this section, we propose solutions to several interesting problems using our privacy-preserving sum and product calculation protocols.

8.1 Polynomial Evaluation

Our protocols can be used to evaluate some polynomials of private values from participants $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ (Eq. 3) as in our conference version [29].

$$f(\mathbf{x}) = \sum_{k=1}^{m} (c_k \prod_{i=1}^{n} x_i^{d_{i,k}}) \tag{3}$$

Here the coefficients and powers are publicly known. However, we are not able to extend our protocols to obliviously evaluate any form of multivariate polynomials yet because directly using our protocols will reveal the value of every product term $\prod x_i^{d_{i,k}}$ ([29]).

8.2 Statistics Calculation

Privacy-preserving statistics calculation is desired in various applications [8], [18], [1]

Our protocols can be directly used to express various statistical values. For example, $\sum_{i=1}^{n} x_i$ is directly computed by our sum protocol, and thus the mean $\mu = \sum_{i=1}^{n} x_i/n$ can be computed with privacy preservation. Given the mean μ, the following can be achieved as well:

$$n\mu^2 + \sum_{i=1}^{n} (x_i^2 - 2x_i\mu)$$

This divided by n is the population variance. Similarly, other statistical values can also me computed with privacy preservation (e.g., *sample skewness*,*k-th moment*, *mean square weighted deviation*, *regression*, or *randomness* test) based on our sum and product protocols.

8.3 General Boolean Formula Evaluation

A boolean formula consists of True or False variables and logical operators (AND, OR, XOR etc.). It is important to securely evaluation a boolean formula in many problems (Multi-party Millionaire problem, Anonymous voting problem), and we show how to achieve a general boolean formula evaluation without disclosing individuals' input values. In our protocol, x_i's are numeric values, so we define $\mathbf{p}_i$'s True-False variable $X_i = T$ if $x_i = 1$ and $X_i = F$ if $x_i = 0$, and we consider all other values as invalid input. Then, we construct the following conversion table:

Note that it is extensively discussed in previous works about the transformation from the polynomial formula

TABLE 6
Conversion Table from boolean formula to polynomial formula

Boolean Formula	Polynomial Formula
$\neg X_i$	$1 - x_i$
$X_i \vee X_j$	$1 - (x_i - 1)^2(x_j - 1)^2$
$X_i \wedge X_j$	$x_i x_j$
$X_i \oplus X_j$	$(x_i - x_j)^2$

to the boolean formula or vice versa (e.g., [30], [35], [13]), and ours is just one of the feasible solutions. With Table 6, we can convert arbitrary boolean formula to the polynomial having equivalent value ($T \to 1, F \to 0$). For example, $X_i \to X_j$ can be converted to $\neg X_i \vee X_j$ first and then be converted to the polynomial:

$$\begin{aligned} & 1 - ((1 - x_i) - 1)^2(x_j - 1)^2 \\ =& 1 - x_i^2(x_j - 1)^2 \\ =& - x_i^2 x_j^2 + x_i^2 \cdot 2x_j - x_i^2 + 1 \end{aligned}$$

Then, our multivariate polynomial evaluation protocol can be run once for each boolean formula to evaluate the final value securely. However, note that the number of participants should be at least three in participants only model and two in aggregator only model due to the issue mentioned in Section 6.1.

8.4 Veto Protocol

A privacy-preserving veto protocol requires that only the final outcome (whether there exists any veto) is published without disclosing any individual vote. This can be easily implemented by employing the above boolean formula evaluation as a building block.

Firstly, we let $X_i = T$ when $\mathbf{p}_i$ vetoes and $X_i = F$ otherwise. Then, the following boolean formula tells whether there is any veto:

$$\bigvee_{i=1}^{n} X_i$$

which can be converted to the polynomial formula:

$$v = 1 - \prod_{i=1}^{n} (x_i - 1)^2$$

where $x_i = 1$ if $\mathbf{p}_i$ vetoes and $x_i = 0$ otherwise. Then, the outcome of the polynomial formula $v = 1$ if anyone vetoes and $v = 0$ otherwise. This polynomial can be evaluated securely using our protocol only once.

8.5 Millionaire Problem

The traditional millionaire problem requires to tell who is the richer between two millionaire while neither party knows the exact amount of money the other party possesses. This problem seems impossible to solve using our protocol since we require at least 3 participants in our participants only model, but we can solve this with a simple trick.

First, we represent two millionaires' money (assumed to be integer for simplicity) b_1, b_2 in a binary format $\{0,1\}^k$. That is, $b_i = \sum_{j=1}^{k} 2^i b_{ij}$. Then, if we convert the $b_{ij} = 1$ to $B_{ij} = T$ and $b_{ij} = 0$ to $B_{ij} = F$, we have the following boolean formula to compute the millionaire problem as in [16]:

$$M = \bigvee_{j=1}^{k} (B_{1j} \wedge \neg B_{2j} \wedge \bigwedge_{l=j+1}^{k} (\neg(B_{1d} \oplus B_{2d})))$$

where $M = T$ indicates $b_1 > b_2$ and $M = F$ indicates $b_1 \leq b_2$.

This can be converted to the polynomial formula as mentioned above, and our polynomial evaluation protocol can be executed only once to evaluate the final result.

Although we only have two participants, but each one's input is divided into k binary inputs, therefore our protocol can be used.

8.6 Maximum (Minimum) Function

Finding the maximum or minimum value without knowing its data provider is important in many data mining applications. This function can be implemented using our protocol with some special conversion. Let $\{1, 2, \cdots, v\}$ be the set of possible input values, and we present two approaches to find the $\max(x_1, x_2, \cdots, x_n)$ or $\min(x_1, x_2, \cdots, x_n)$.

Linear Solution

If we compute the following term for every $i \in \{1, 2, \cdots, v\}$:

$$M_i = \prod_{j=1}^{n} (x_j - i)^2$$

Then, $M_i = 0$ if and only if there exists at least one $x_j = i$. Therefore, the largest (smallest) i makes $M_i = 0$ is the largest (smallest) value among $\{x_1, \cdots, x_n\}$. Then, one needs to request and evaluate the M_i starting from $i = v$ to $i = 1$ ($i = 1$ to $i = v$) until he finds the first i makes $M_i = 0$ to find the maximum (minimum) value. However, our polynomial evaluation protocol should be run v times in the worst case (i.e., the communication complexity is $O(v)$).

Sublinear Solution

We further improve the complexity by using the Veto Protocol above as a building block. One guesses a number x as the maximum value and asks every participant whether someone has a larger (smaller) value, any participant having a larger (smaller) value vetoes to the request. No one vetoes at x does not mean x is the maximum (minimum). We need to find an x until we are sure that someone vetoes at $x - 1$ ($x + 1$) but no one vetoes at x.

The actual algorithm is described as follows:

Algorithm 1 Binary Maximum (Minimum) Search

1: Set the boundaries: $l := 1, r := v$, and set the initial guess
2: . x is guessed as the maximum (minimum) value, and the Veto Protocol is initiated.
3: Any participant vetoes if his value is larger (smaller) than x.
4: If $l = r - 1$, return x as the maximum (minimum) value.
If someone vetoes at x, $l := x$ ($r := x$) and let $x := \lceil \frac{l+r}{2} \rceil$ ($x := \lfloor \frac{l+r}{2} \rfloor$). Then, re-start from 2.
If no one vetoes at x, $r := x$ ($l := x$) and let $x := \lfloor \frac{l+r}{2} \rfloor$ ($x := \lceil \frac{l+r}{2} \rceil$). Then, re-start from 2.

The algorithm ends always after $\lceil \log v \rceil$ rounds of the searches because the algorithm returns the value only when $l = r$. Therefore, our protocol should be run $\lceil \log v \rceil$ times.

According to the actual application, either participants only model or one aggregator model can be used in the secure polynomial evaluation protocol. However, if we use the one aggregator model and we do not want to let participants know which number is the maximum (minimum) value, we have to use the **Linear Solution**. In the **Linear Solution**, the aggregator can send dummy requests after finding the maximum (minimum) value to hide the value, but in the **Sublinear Solution**, participants immediately know the maximum (minimum) value after the $\lceil \log v \rceil$ rounds of the requests. Unlike the **Linear Solution**, no further dummy requests are possible since the search space is already narrowed down such that only one candidate is left.

Yet, it is still possible to have a safer sublinear solution. Essentially, finding the maximum (minimum) value from $\{1, \cdots, v\}$ via binary search is finding a path leading to the maximum value in a balanced binary search tree having v leaf nodes. Then, the aggregator may randomly find $C-1$ numbers and launch the fake binary search to find those $C - 1$ random numbers. Then, participants only infer that the true maximum number is one of the C numbers, i.e., the probability to infer the correct maximum number is $\frac{1}{C}$. Then, the number of rounds that our protocol should be run is $C \cdot \lceil \log v \rceil = O(\log v)$, which is still sublinear.

8.7 Sealed-Bid First-Price Auction

We consider a sealed-bid 1st-price auction with single object. In such an auction, multiple bidders place their bids for an object and the bidder with the highest bid gets the object, and the amount of cost he pays is his bid. We employ the above privacy-preserving maximum function as a building block to implement a privacy preserving 1st-price auction.

In general, the bid information is confidential and kept secret to the bidders, so we employ the one aggregator model here and let the auctioneer act as the aggregator in the polynomial evaluation protocol (who is the only party that collects the final result).

First, we use the above function $\max(x_1, x_2, \cdots, x_n)$ to find the maximum value (say max). Since we employ the one aggregator model in the protocol, only the auctioneer knows the maximum value max. Then, we design a simple **Anonymous Exchange Protocol** which informs the auctioneer whether $x_i = max$ for each $\mathbf{p}_i$ while max is kept secret to all $\mathbf{p}_i$'s and the auctioneer do not know participants' input data except the one belong to those who possess the maximum value. Note that we do not require a secure channel in the following protocol either.

Algorithm 2 Anonymous Exchange Protocol

1: The auctioneer finds a large random number (not necessarily prime) $R \gg v^2$ and keeps it secret.
2: The auctioneer computes $t_A = R - max$ and sends t_1 to all participants $\mathbf{p}_i$'s.
3: Each participant $\mathbf{p}_i$, after receiving t_A, computes $t_i = (t_A + x_i)c_i$ where c_i is a secret random number picked from $\mathbb{Z}_v$. Then, each participant sends t_i to the auctioneer.
4: After receiving t_i's, the auctioneer computes $k_i = t_i \mod R$. $k_i = 0$ if $x_i = max$, and it is a random number if $x_i \neq max$.

In the above algorithm, t_A is masked by the R, and the x_i in every t_i is masked by the c_i. Therefore the participants do not know max and the auctioneer do not know x_i's either (except the maximum x_i). Therefore, the auctioneer knows who placed the highest bid. If there exist multiple highest bidder, some mechanism is required to choose the final winner.

8.8 Sealed-Bid Second-Price Auction

In the 2nd-price auction, the highest bidder's cost is not his bid. Instead, his payment is same as the next highest bid. This type of auction is widely used because bidders achieves maximum benefit if and only if they bid truthfully [12]. After finding the maximum value using the maximum function above, we can set $r := max - 1$ to find the next largest value. Then, it seems trivial to implement the 2nd-price auction by running the maximum function twice to get the 2nd-highest bid. This is feasible only when there is no tie in the auction. If multiple bidders place the same highest bid, the next largest value found by the maximum function above is not the next highest bid. The next highest bid, instead, should be same as the highest bid (due to duplicate bids). Therefore, we need another method to find the next highest bid. Since the payment of the winner is not his bid, the auction should be composed of two parts: winner determination & payment determination.

Winner Determination

The winner of the 2nd-price auction is the person whose bid is the highest among the participants. Therefore, we use the aforementioned Maximum Function to find the value of the highest bid. Then, we run the aforementioned **Anonymous Exchange Protocol** to determine the highest bidder(s). By this protocol, the auctioneer not only knows who are the highest bidders, but also know the number of highest bidders.

Payment Determination

After the winner determination, if there are multiple winners, the auctioneer introduces any mechanism (depending on the auction type) to determine the final winner, and the payment of the winner is just the highest value found during the winner determination. If there is only one winner, we can find the 2nd highest bid in the same way as we mentioned at the beginning of this section, which is the winner's payment. In both cases, the number of times we need to run our protocol is $O(\log v)$.

In conclusion, the number of times we need to run our protocol for each application is summarized in the following table:

TABLE 7
Communication Complexity Comparison

Application	Number of times
Boolean Formula Evaluation	$O(1)$
Veto Protocol	$O(1)$
Millionaire Problem	$O(1)$
Maximum (Minimum) Function	$O(\log v)$ (Sublinear Solution)
Sealed-Bid First-Price Auction	$O(\log v)$ (Sublinear Solution)
Sealed-Bid Second-Price Auction	$O(\log v)$ (Sublinear Solution)

9 Conclusion

In this paper, we successfully achieve privacy-preserving sum and product calculation protocols without secure communication channels or trusted key issuers. We allow up to k (adjustable parameter) collusive participants who will not tamper the computation but try to manipulate their parameters to infer others' private values. We formally analyzed the security of our protocols and showed that the protocols are secure if the CDH problem is assumed to be intractable, and we also showed with implementation that the protocols are efficient to be applicable in real life. At the end, we propose numerous applications that are achieved from our protocols.

One of our future works is to design privacy preserving data releasing protocols such that general function of data can be evaluated correctly while preserving individuals' data privacy.

Acknowledgement

This paper would have not existed without the insightful reviews and constructive suggestions. The precious comments greatly inspired us, and they also brought considerable improvements and corrections to this paper. We sincerely appreciate all of the reviews and all the reviewers who were very responsible and professional.

References

[1] C. C. Aggarwal and S. Y. Philip, *A general survey of privacy-preserving data mining models and algorithms*. Springer, 2008.

[2] D. Beaver, S. Micali, and P. Rogaway, "The round complexity of secure protocols," in *STOC*. ACM, 1990, pp. 503–513.

[3] A. Ben-David, N. Nisan, and B. Pinkas, "Fairplaymp: a system for secure multi-party computation," in *CCS*. ACM, 2008, pp. 257–266.

[4] D. Boneh, "The decision diffie-hellman problem," in *Algorithmic number theory*. Springer, 1998, pp. 48–63.

[5] C. Castelluccia, A. C. Chan, E. Mykletun, and G. Tsudik, "Efficient and provably secure aggregation of encrypted data in wireless sensor networks," *TOSN*, vol. 5, no. 3, p. 20, 2009.

[6] C. Castelluccia, E. Mykletun, and G. Tsudik, "Efficient aggregation of encrypted data in wireless sensor networks," in *MobiQuitous*. IEEE, 2005, pp. 109–117.

[7] T.-H. H. Chan, E. Shi, and D. Song, "Privacy-preserving stream aggregation with fault tolerance," in *Financial Cryptography and Data Security*. Springer, 2012, pp. 200–214.

[8] X. Chen, X. Wu, X.-Y. Li, Y. He, and Y. Liu, "Privacy-preserving high-quality map generation with participatory sensing," in *INFOCOM*. IEEE, 2014.

[9] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu, "Tools for privacy preserving distributed data mining," *SIGKDD Explorations Newsletter*, vol. 4, no. 2, pp. 28–34, 2002.

[10] C. Dwork, "Differential privacy," in *Automata, languages and programming*. Springer, 2006, pp. 1–12.

[11] C. Dwork and J. Lei, "Differential privacy and robust statistics," in *STOC*. ACM, 2009, pp. 371–380.

[12] B. Edelman, M. Ostrovsky, and M. Schwarz, "Internet advertising and the generalized second price auction: Selling billions of dollars worth of keywords," National Bureau of Economic Research, Tech. Rep., 2005.

[13] B. J. Falkowski, "A note on the polynomial form of boolean functions and related topics," *Transactions on Computers*, vol. 48, no. 8, pp. 860–864, 1999.

[14] N. Fazio, R. Gennaro, I. M. Perera, and W. E. Skeith III, "Hard-core predicates for a diffie-hellman problem over finite fields," in *CRYPTO*. Springer, 2013, pp. 148–165.

[15] J. Feigenbaum and M. Merritt, *Distributed computing and cryptography: proceedings of a DIMACS Workshop, October 4-6, 1989*. American Mathematical Soc., 1991, vol. 2.

[16] M. Fischlin, "A cost-effective pay-per-multiplication comparison method for millionaires," in *Topics in CryptologyCT-RSA 2001*. Springer, 2001, pp. 457–471.

[17] A. Friedman and A. Schuster, "Data mining with differential privacy," in *SIGKDD*. ACM, 2010, pp. 493–502.

[18] B. Fung, K. Wang, R. Chen, and P. S. Yu, "Privacy-preserving data publishing: A survey of recent developments," *ACM Computing Surveys (CSUR)*, vol. 42, no. 4, p. 14, 2010.

[19] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *STOC*. ACM Press, 2009, pp. 169–169.

[20] C. Gentry and S. Halevi, "Implementing gentrys fully-homomorphic encryption scheme," in *EUROCRYPT*. Springer, 2011, pp. 129–148.

[21] O. Goldreich, "Secure multi-party computation," *Manuscript. Preliminary version*, 1998.

[22] S. Goryczka, L. Xiong, and V. Sunderam, "Secure multiparty aggregation with differential privacy: A comparative study," in *Proceedings of the Joint EDBT/ICDT 2013 Workshops*. ACM, 2013, pp. 155–163.

[23] W. He, X. Liu, H. Nguyen, K. Nahrstedt, and T. Abdelzaher, "Pda: Privacy-preserving data aggregation in wireless sensor networks," in *INFOCOM*. IEEE, 2007, pp. 2045–2053.

[24] M. Jawurek and F. Kerschbaum, "Fault-tolerant privacy-preserving statistics," in *Privacy Enhancing Technologies*. Springer, 2012, pp. 221–238.

[25] M. Joye and B. Libert, "A scalable scheme for privacy-preserving aggregation of time-series data," in *Financial Cryptography and Data Security*. Springer, 2013, pp. 111–125.

[26] T. Jung and X.-Y. Li, "Infinite choices of data aggregations with linear number of keys," *arXiv preprint arXiv:1308.6198*, 2013.

[27] T. Jung, X.-Y. Li, Z. Wan, and M. Wan, "Privacy preserving cloud data access with multi-authorities," in *INFOCOM*. IEEE, 2013, pp. 2625–2633.

[28] T. Jung, X.-Y. Li, and L. Zhang, "A general framework for privacy-preserving distributed greedy algorithm," *arXiv preprint arXiv:1307.2294*, 2013.

[29] 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 *INFOCOM*. IEEE, 2013, pp. 2634–2642.

[30] J. Katz, A. Sahai, and B. Waters, "Predicate encryption supporting disjunctions, polynomial equations, and inner products," in *EUROCRYPT*. Springer, 2008, pp. 146–162.

[31] Q. Li and G. Cao, "Efficient privacy-preserving stream aggregation in mobile sensing with low aggregation error," in *Privacy Enhancing Technologies*. Springer, 2013, pp. 60–81.

[32] X.-Y. Li and T. Jung, "Search me if you can: privacy-preserving location query service," in *INFOCOM*. IEEE, 2013, pp. 2760–2768.

[33] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*. CRC press, 1996.

[34] M. Naehrig, K. Lauter, and V. Vaikuntanathan, "Can homomorphic encryption be practical?" in *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*. ACM, 2011, pp. 113–124.

[35] W. G. Schneeweiss, "On the polynomial form of boolean functions: derivations and applications," *IEEE Transactions on Computers*, vol. 47, no. 2, pp. 217–221, 1998.

[36] R. Sheikh, B. Kumar, and D. Mishra, "Privacy preserving k secure sum protocol," *Arxiv preprint arXiv:0912.0956*, 2009.

[37] E. Shi, T.-H. H. Chan, E. G. Rieffel, R. Chow, and D. Song, "Privacy-preserving aggregation of time-series data." in *NDSS*, vol. 2, 2011, p. 4.

[38] V. S. Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, and Y. Theodoridis, "State-of-the-art in privacy preserving data mining," *SIGMOD*, vol. 33, no. 1, pp. 50–57, 2004.

[39] A. C.-C. Yao, "Protocols for secure computations," in *FOCS*, vol. 82, 1982, pp. 160–164.

[40] ——, "How to generate and exchange secrets," in *FOCS*. IEEE, 1986, pp. 162–167.

[41] L. Zhang, X.-Y. Li, and Y. Liu, "Message in a sealed bottle: Privacy preserving friending in social networks," in *ICDCS*. IEEE, 2013, pp. 327–336.

[42] L. Zhang, X.-y. Li, Y. Liu, and T. Jung, "Verifiable private multi-party computation: ranging and ranking," in *INFOCOM*. IEEE, 2013, pp. 605–609.

[43] J. Zhao, T. Jung, Y. Wang, and X.-Y. Li, "Achieving differential privacy of data disclosure in the smart grid," in *INFOCOM*. IEEE, 2014.

Taeho Jung received the B.E degree in Computer Software from Tsinghua University, Beijing, in 2007, and he is working toward the Ph.D degree in Computer Science at Illinois Institute of Technology while supervised by Dr. Xiang-Yang Li. His research area, in general, includes privacy & security issues in mobile network and social network analysis. Specifically, he is currently working on the privacy-preserving computation in various applications and scenarios.

Dr. Xiang-Yang Li is a professor at the Illinois Institute of Technology, and an EMC Endowed Visiting Chair Professor (2013-2016), Tsinghua University. He is a recipient of China NSF Outstanding Overseas Young Researcher (B). Dr. Li received MS (2000) and PhD (2001) degree at Department of Computer Science from University of Illinois at Urbana-Champaign, a Bachelor degree at Department of Computer Science and a Bachelor degree at Department of Business Management from Tsinghua University, P.R. China, both in 1995. His research interests include mobile computing, cyber physical systems, wireless networks, security and privacy, and algorithms. He published a monograph "Wireless Ad Hoc and Sensor Networks: Theory and Applications". He co-edited several books, including, "Encyclopedia of Algorithms". Dr. Li is an editor of several journals, including IEEE Transaction on Parallel and Distributed Systems, IEEE Transaction on Mobile Computing. He has served many international conferences in various capacities, including ACM MobiCom, ACM MobiHoc, ACM STOC, IEEE MASS, and so on. He is a senior member of IEEE and a member of ACM. The research of Xiang-Yang Li is partially supported by USA NSF, National Natural Science Foundation of China, and RGC of HongKong.

Meng Wan is a professor at Center for Science and Technology Development at Ministry of Education. He received his Ph. D. from Wuhan University in 2008, and his M.S. from Central University of Finance and Economics in 2000. His research interests include computer network architecture, network and systems management, science and technology management, system engineering etc. He is currently serving as the division director of Department of Network and Information in Center for Science and Technology Development at Ministry of Education and the associate editor of Sciencepaper Online. He is a member of the IEEE,ACM.