

Privacy-Preserving Data Aggregation without Secure Channel: Multivariate Polynomial Evaluation

Taeho Jung[†], XuFei Mao[‡], Xiang-Yang Li[†], Shao-Jie Tang[†], Wei Gong[‡], and Lan Zhang[‡]

[†]Department of Computer Science, Illinois Institute of Technology, Chicago, IL

[‡]School of Software, TNLIST, Tsinghua University, 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 propose several protocols that successfully guarantee data privacy under this weak assumption while limiting both the communication and computation complexity of each participant to a small constant.

Index Terms—Privacy, aggregation, secure channels, SMC, homomorphic.

I. 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 entire 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 individual’s 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 this problem.

Secure Multi-party Computation (SMC) was first formally introduced by Yao [22] in 1982 as Secure Two-Party Computation. Generally, it enables n parties who want to jointly and

privately compute a function

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

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 this approach. However this actually does not completely solve our problem because interactive invocation is required for participants in synchronous SMC (e.g., [13]), which leads to high communication and computation complexity, which will be compared in the Section VIII. 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. That is, the operator who executes homomorphic operations upon the ciphertexts are not authorized to achieve the final result. This forbids aggregator from decrypting the aggregated result, because if the aggregator is allowed to decrypt the final result, he can also decrypt the individual ciphertext received, which contradicts our motivation. Also, because the size of the plaintext space is limited, the number of addition and multiplication operations executed upon ciphertexts was limited until Gentry *et al.* proposed a fully homomorphic encryption scheme [11] and implemented it in [12]. However, Lauter *et al.* pointed out in [16] that the complexity of general HE is too high to use in real application. Lauter also proposed a HE scheme which sacrificed possible number of multiplications for speed, but it still needs too much time to execute homomorphic operations on ciphertexts.

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 hole since the security of those schemes relies on the assumption that keys are disclosed to authorized participants

¹The research of authors is partially supported by NSFC under Grant No. 61170216, No. 61228202, and No. 61272426, China 973 Program under Grant No.2011CB302705, China Postdoctoral Science Foundation funded project under grant No. 2012M510029, NSF CNS-0832120, NSF CNS-1035894, NSF ECCS-1247944.

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 multiplications is restricted
Cons	decrypter can decrypt both aggregated data and individual data trade-off between # of multiplications and complexity exists

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 and secure pair-wise communication channels. The main contributions of this paper are:

- *Formulation of a model without secure channel*: Different from many other models in privacy-preserving data aggregation problem, our model does not require a secure communication channel throughout the protocol.
- *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.
- *General Multivariate Polynomial Evaluation*: We generalize the privacy-preserving data aggregation to secure multivariate polynomial evaluation whose inputs are jointly provided by multiple parties. That is, our scheme enables multiple parties to securely compute

$$f(\{x_1, \dots, x_n\}) = \sum_{k=1}^m c_k \left(\prod_{i=1}^n x_i^{d_{i,k}} \right)$$

where the data x_i is a privately known data by user i .

Note that our general format of data aggregation can be directly used to express various statistical values. For example, $\sum_{i=1}^n x_i$ can easily be achieved while preserving privacy, and thus the mean $\mu = \sum_{i=1}^n x_i/n$ can be computed with privacy-preserving. Given the mean μ , $n\mu^2 + \sum_{i=1}^n (x_i^2 - 2x_i\mu)$ can be achieved from the polynomial, and this divided by n is the population variance. Similarly, other statistical values are also achievable (e.g., *sample skewness, k-th moment, mean square weighted deviation, regression, and randomness test*) based on our general multi-variate polynomial. Although our methods are proposed for computing the value of a multi-variate polynomial function where the input of each participant is assumed to be an integer, our methods can be generalized for functions (such as dot product) where the input of each participant is a vector.

The rest of the paper is organized as follows. We present the system model and necessary background in Section III. In Section IV, we analyze the needed number of communications with secure communication channels when users communicate randomly. We first address the privacy preserving summation and production in Section V by presenting two efficient protocols. Based on these protocols, we then present an efficient protocol for general multi-variate polynomial evaluation in

Section VI. In Section VII, we present detailed analysis of the correctness, complexity, and security of our protocols. Performance evaluation of our protocols is reported in Section VIII. We compare our protocol with the ones based on SMC or HE. We then conclude the paper with the discussion of some future work in Section IX.

II. 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 could decrypt each individual sensor's data, and they assumed the trusted aggregator in their model.

Sheikh *et al.* [19] proposed a k -secure sum protocol, which is motivated by the work of Clifton *et al.* [7]. They significantly reduced the probability of data leakage in [7] 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. Similar to our protocol, one can combine their sum protocol and converted product protocol to achieve a privacy-preserving multivariate polynomial evaluation protocol. 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.* [14] 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.* [20] 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 to the multivariate polynomial evaluation problem, 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 $\prod_{i=0}^n k_i = 1$, 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 $i = 1, 2, \dots, n$). Then, the aggregator can recover the sum $\sum_{i=1}^n x_i$ iff he received ciphertexts from all of the participants. He computes $k_0 \prod_{i=1}^n C_i$ 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 [18] 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 .

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 [14][19], 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 [20], but we do not trivially employ brute-force manner in decryption, instead, we employ our novel efficient protocols for sum and product calculation.

III. SYSTEM MODELS AND PRELIMINARY

A. System Model and Problem Definition

Assume that there are n participants $\{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_n\}$, and each participant \mathbf{p}_i has a privately known data x_i from a group \mathbb{G}_1 . The privacy-preserving data aggregation problem (or secure multivariate polynomial evaluation problem) is to compute some multivariate polynomial of x_i jointly or by an aggregator while preserving the data privacy. Assume that there is a group of m powers $\{d_{i,k} \in \mathbb{Z}_q \mid k = 1, 2, \dots, m\}$ for each \mathbf{p}_i and m coefficients $\{c_k \mid k = 1, \dots, m, c_k \in \mathbb{G}_1\}$. The objective of the aggregator or the participants is to compute the following polynomial without knowing any individual x_i :

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

Here vector $\mathbf{x} = (x_1, x_2, \dots, x_n)$. For simplicity, we assume that the final result $f(\mathbf{x})$ is positive and bounded from above by a large prime number P . We assume all of the powers $d_{i,k}$'s and coefficients c_k 's are open to any participant as well as the attackers. This is a natural assumption since the powers and coefficients uniquely determine a multivariate polynomial, and the polynomial is supposed to be public.

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. We also assume participants do not trust each other and that they are curious as well,

however, they will follow the protocol in general. 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 call this model the *Participants Only Model*.

In both models, participants are assumed not to collude with each other. Relaxing this assumption is one of our future work.

B. Additional Assumptions

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 discrete logarithm problem is computationally hard if: 1) the orders of the integer groups are large prime numbers; 2) the involved integer numbers are large numbers. The security of our scheme relies on this assumption. We further assume that there is a secure pseudorandom function (PRF) which can choose a random element from a group such that this element is computationally indistinguishable to uniform random.

We also assume that user authentication was in place to authenticate each participants if needed. We note that Dong *et al.* [9] investigated verifiable privacy-preserving dot production of two vectors and Zhang *et al.* [24] proposed verifiable multi-party computation, both of which can be partially or fully exploited later. Designing privacy preserving data aggregation while providing verification of the correctness of the provided data is a future work.

C. Discrete Logarithm Problem

Let $\mathbb{G} \subset \mathbb{Z}_p$ be a cyclic multiplicative integer group, where p is a large prime number, and g be a generator of it. Then, for all $h \in \mathbb{G}$, h can be written as $h = g^k$ for some integer k , and any integers are congruent modulo p . The discrete logarithm problem is defined as follows: given an element $h \in \mathbb{G}$, find the integer k such that $g^k = h$.

The famous Decision Diffie-Hellman (DDH) problem proposed by Diffie and Hellman in [8] is derived from this assumption. DDH problem is widely exploited in the field of cryptography (e.g., El Gamal encryption [10] and other cryptographic security protocols such as CP-ABE [3]) as discussed in [4]. Our protocol is based on the assumption that it is computational expensive to solve the discrete logarithm problem as in other similar research works ([15], [17], [24]).

IV. ACHIEVING SUM UNDER SECURED COMMUNICATION 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.* [7], which is also reviewed in [21], but we found their setting imposed unnecessary communication overhead, and we reduced it while maintaining the same security level. Assume participants $\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_n$ 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 IV.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 IV.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.

V. EFFICIENT PROTOCOLS FOR SUM AND PRODUCT

In this section, we present two novel calculation protocols for each model which preserve individual's data privacy. These four protocols will serve as bases of our solution to privacy-preserving data aggregation problem. For simplicity, we assume all coefficients c_k ($k \in [1, m]$) and powers $d_{i,k}$ ($i \in [1, n], k \in [1, m]$) of the polynomial $f(\mathbf{x}) = \sum_{k=1}^m c_k (\prod_{i=1}^n x_i^{d_{i,k}})$ are known to every participant \mathbf{p}_i . Table I summarizes the main notations used in this paper.

TABLE I
NOTATIONS OF SYMBOLS USED IN OUR PROTOCOLS

\mathbf{p}_i	i -th participant in data aggregation
\mathcal{A}	Aggregator
$\mathbb{G}_1, \mathbb{G}_2$	multiplicative cyclic integer groups
g_1, g_2	generators of above groups
$d_{i,k}$	power of $x_i^{d_{i,k}}$
c_k	coefficient of $c_k \sum_{i=1}^n x_i^{d_{i,k}}$
r_i, \hat{r}_i	randomly chosen numbers

A. Product Protocol - Participants Only Model

Firstly, we assume that all participants together want to compute the value $f(\mathbf{x}) = \prod_i x_i$ given their privately known values $x_i \in \mathbb{Z}_p$. The basic idea of our protocol is to find some random integers $R_i \in \mathbb{Z}_p$ such that $\prod_i R_i = 1 \pmod p$ 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 .

Let $\mathbb{G}_1 \subset \mathbb{Z}_p$ be a cyclic multiplicative group of prime order p and g_1 be its generator. Then our protocol for privacy preserving production $\prod_i x_i$ has the following steps: **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 ring for computation purpose. The ring can be formed according to the lexicographical order of the MAC address or even the geographical location. It is out of our scope to consider this problem. Each $\mathbf{p}_i (i \in \{1, \dots, n\})$ randomly chooses a secret integer $r_i \in \mathbb{Z}_q$ using PRF and calculates a public parameter $g_1^{r_i} \in \mathbb{G}_1$. Then, each \mathbf{p}_i shares $Y_i = g_1^{r_i} \pmod p$ with \mathbf{p}_{i-1} and \mathbf{p}_{i+1} (here $\mathbf{p}_{n+1} = \mathbf{p}_1$ and $\mathbf{p}_0 = \mathbf{p}_n$).

After a round of exchanges, the participant \mathbf{p}_i computes the number $R_i = (Y_{i+1}/Y_{i-1})^{r_i} = (g_1^{r_{i+1}}/g_1^{r_{i-1}})^{r_i} \pmod p$ and

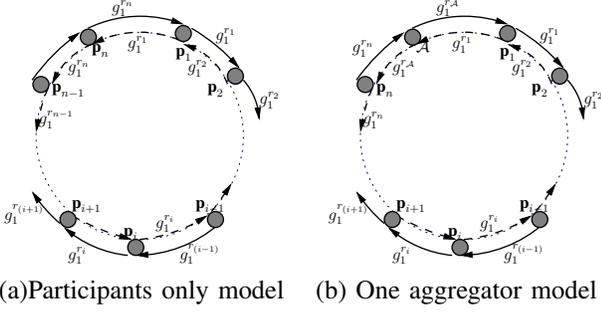


Fig. 1. Communications in Setup

keeps this number R_i secret. Note \mathbf{p}_1 calculates $(g_1^{r_2}/g_1^{r_n})^{r_1}$ and \mathbf{p}_n calculates $(g_1^{r_1}/g_1^{r_{(n-1)}})^{r_n}$.

Encrypt(x_i) $\rightarrow C_i \in \mathbb{G}_1$

When a product is needed, every \mathbf{p}_i creates the ciphertext:

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

where x_i is his private input data. If he does not want to participate in the multiplication, he can simply set $x_i := 1$. Then, he broadcasts this ciphertext.

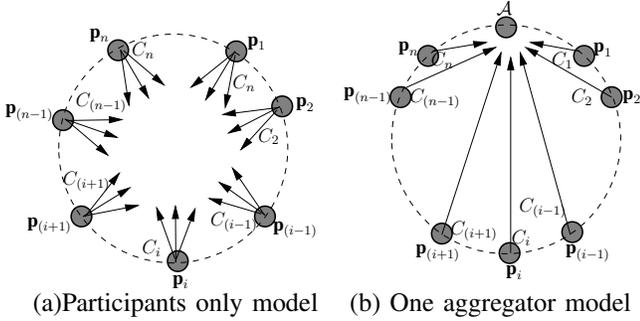


Fig. 2. Communications in Encrypt

Product($\{C_1, C_2, \dots, C_n\}$) $\rightarrow \prod_{i=1}^n x_i \in \mathbb{G}_1$

Any \mathbf{p}_i , after receiving n ciphertexts $\{C_1, C_2, \dots, C_n\}$ from all of the \mathbf{p}_i 's, calculates the following product:

$$\prod_{i=1}^n C_i = \prod_{i=1}^n x_i \pmod{p}$$

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

B. Product Protocol - One Aggregator Model

We use the same group used in Participants Only Model. Everything is same as the protocol above, except that the aggregator \mathcal{A} acts as the $(n+1)$ -th participant \mathbf{p}_{n+1} . In other words, there are $n+1$ "participants" now. 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. 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 regular participants.

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

$$(g_1^{r_1}/g_1^{r_n})^{r_{n+1}} \prod_{i=1}^n x_i = \prod_{i=1}^n x_i$$

to achieve the final product, where r_{n+1} is the random number generated by \mathcal{A} .

C. Sum Protocol - Participants Only Model

Here we assume that all participants together want to compute the value $f(\mathbf{x}) = \sum_{i=1}^n x_i$ given their privately known values $x_i \in \mathbb{Z}_p$. It seems that we can still exploit the method used for computing product by finding random numbers R_i such that $\sum_{i=1}^n R_i = 0$. We found that it is challenging to find such a number R_i while preserve privacy and security. The basic idea of our protocol is to convert the sum of numbers into production of numbers. Previous solution [20] essentially applied this approach also by computing the product of $\prod_{i=1}^n g^{x_i} = g^{\sum_{i=1}^n x_i}$. Then find $\sum_{i=1}^n x_i$ by computing the discrete logarithm of the product. As discrete logarithm is computational expensive, we will not adopt this method. Instead, we propose a computational efficient method here.

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

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

From the Equation (2), we conclude that

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

Our protocol works as follows. Let $\mathbb{G}_2 \subset \mathbb{Z}_{p^2}$ be a cyclic multiplicative group of order $p(p-1)$ and g_2 be its generator, where p is a prime number. Then our protocol for privacy preserving summation $\prod_i x_i$ has the following steps: Setup, Encrypt, Sum.

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

Remember that participants are arranged in a circle. \mathbf{p}_i uses PRF to randomly pick a secret number $r_i \in \mathbb{Z}_{pq}$, and calculates a public parameter $g_2^{r_i}$. Then, he shares $g_2^{r_i}$ 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 $R_i = (g_2^{r_{i+1}}/g_2^{r_{i-1}})^{r_i}$ and keeps this secret.

Encrypt(x_i, R_i) $\rightarrow C_i \in \mathbb{G}_2$

This algorithm crosses over two different integer groups: \mathbb{G}_1 and \mathbb{G}_2 . Each \mathbf{p}_i first calculates $(1+x_i \cdot p) \pmod{p^2}$. Note that $x_i \in \mathbb{G}_1$, and it is temporarily treated as an element in \mathbb{G}_2 , but this does not affect the last value of the result since operations in \mathbb{G}_2 are modulo p^2 . 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:

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

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

$$\text{Sum}(\{C_1, C_2, \dots, C_n\}) \rightarrow \sum_{k=1}^n x_i \in \mathbb{G}_1.$$

Each participant, after receiving the ciphertexts from all of other participants, calculates the following $C \in \mathbb{G}_2$:

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

Then, he calculates $(C - 1)/p \pmod{p} = \sum_{i=1}^n x_i \pmod{p}$ to recover the final sum.

D. Sum Protocol - One Aggregator Model

Similar to the product protocol for One Aggregator Model, everything is the 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

$$C = (g_2^{r_1}/g_2^{r_n})^{r_{n+1}} \prod_{i=1}^n C_i = (1 + p \sum_{i=1}^n x_i) \pmod{p^2}$$

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

VI. EFFICIENT PROTOCOLS FOR GENERAL MULTIVARIATE POLYNOMIAL

Now we are ready to present our efficient privacy preserving protocols for evaluating a multivariate polynomials. Our protocol is based on the efficient protocols for sum and production presented in the previous section.

A. One Aggregator Model

The calculation of the polynomial 1 can be divided into nm multiplications and m additions. In this section we show how to conduct a joint calculation of m products and one sum while preserving individual's data privacy in the One Aggregator Model. Different from the protocols in the Section V, those broadcast ciphertexts are not broadcast this time, they are sent to the aggregator instead. The purpose of this small change is only for reducing communication complexity, and from the security perspective, this is just same as broadcasting since our communication channels are insecure.

1) *Basic Scheme*: All the participants execute **Setup** to initiate the system. Then, for each k , all the participants need to calculate $x_i^{d_{i,k}}$'s first, where $d_{i,k}$'s are powers specified by the aggregator \mathcal{A} , and run the aforementioned product protocol for each $k \in [1, m]$. If \mathcal{A} does not need the data from some participant \mathbf{p}_i , \mathcal{A} can set his powers to be 0, and if \mathbf{p}_i does not want to participate in the aggregation, he can simply set his input as 1.

Then, the aggregator is able to calculate $\sum_{k=1}^m (c_k \prod_{i=1}^n x_i^{d_{i,k}})$.

2) *Advanced Scheme*: The above *Basic Scheme* preserves data privacy in our problem as long as there are at least two $x_i^{d_{i,k}}$'s not equal 1 in each following set $\{x_1^{d_{1,k}}, x_2^{d_{2,k}}, \dots, x_n^{d_{n,k}}\}_{k \in \{1, \dots, m\}}$, which will be further discussed in the Section VII-B1. Therefore, we exploit the aforementioned sum protocol to achieve *Secure Scheme*.

All the participants execute **Setup**. Then, when executing the **Encrypt** of the product protocol, each participant checks whether his input is the only one not equal to 1 for each product $\prod_{i=1}^n x_i^{d_{i,l}}$ (i.e., his $d_{i,l}$ is the only one not equal to 0 in $\{d_{1,l}, d_{2,l}, \dots, d_{n,l}\}$). If it is, the product equals to his input data, which will directly disclose his data, so he skips it. The elements that are omitted form a set $D_{sum} = \{x_i^{d_{i,k}}\}_{k \in I_{sum}}$, where I_{sum} is the set of indices k 's corresponding to the elements in D_{sum} . For each $x_i^{d_{i,k}} \in D_{sum}$, find his owner \mathbf{p}_i and add him into the set P_{sum} . There can be duplicate \mathbf{p}_i 's in the set P_{sum} . The \mathbf{p}_i 's in P_{sum} need to calculate the following without knowing each other's input:

$$\sum_{\mathbf{p}_i \in P_{sum}} c_k x_i^{d_{i,k}}$$

They are called sum participants, and we assume they are ordered by non-decreasing order of their indices in P_{sum} and arranged in a circle. In what follows, we denote \mathbf{p}_i 's successor and predecessor in the P_{sum} as $\mathbf{p}_{i,suc}$ and $\mathbf{p}_{i,pre}$ respectively. These sum participants run the sum protocol to encrypt their data and sends to the aggregator \mathcal{A} .

\mathcal{A} , after receiving all the sum ciphertexts, is able to calculate $\sum_{k \in I_{sum}} c_k x_i^{d_{i,k}}$. Then, he is able to calculate $\sum_{k=1}^m (c_k \prod_{i=1}^n x_i^{d_{i,k}})$.

B. Participants Only Model

From the One Aggregator Model, we know the combination of two protocols (product protocol and second sum protocol) proposed in Section V gives the best scheme. Therefore we only show the scheme which employs both product and sum protocols.

Advanced Scheme: Every participant executes **Setup**, and when he executes the **Encrypt** of the product protocol, he conducts the same examination as in the Section VI-A2 above. Then, the sum participants run the sum protocol to share their sum with each other. Finally, all participants are able to calculate $\sum_{k=1}^m (c_k \prod_{i=1}^n x_i^{d_{i,k}})$ based on others' ciphertexts.

VII. CORRECTNESS, COMPLEXITY AND SECURITY ANALYSIS

Here we provide rigorous correctness proofs, complexity and security analysis of the protocols presented in this paper. We also discuss when our protocols could leak information about the privately known data x_i and provide methods to address this when possible.

A. Correctness

Next we show the correctness of the product protocol in Section V.

1) *Product Protocol*: We only provide the analysis for Participants Only model, but the correctness in One Aggregator model is easily derivable from it. After participants receive $\{C_1, \dots, C_n\}$ they conduct the following calculation:

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

Here $r_{n+1} = r_1, r_0 = r_n$. Thus, the products are correctly calculated.

2) *Sum Protocol*: Similar to above, we only discuss the correctness for Participants Only Model. After participants receive $\{C_1, \dots, C_n\}$, they conduct the following calculation:

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

Thus, $(C - 1)/p \pmod{p}$ is indeed equal to $\sum_{i=1}^n x_i \pmod{p}$.

B. Security

We discuss the security of the schemes in both One Aggregator Model and Participants Only Model in this section.

1) *Special Case of Products Calculation*: As mentioned in the Section VI-A2, if there is only one ciphertext $d_{i,k}$ is not equal to 0 in any set $\{d_{1,k}, d_{2,k}, \dots, d_{n,k}\}_{k \in \{1, \dots, m\}}$ during the products calculation, the individual data x_i can be disclosed to others. This is because:

(suppose that only $d_{i,k}$ is the only ciphertext not equal to 1 in the set $\{d_{1,k}, d_{2,k}, \dots, d_{n,k}\}$)

$$\text{Decrypt}(\{C_{1,k}, C_{2,k}, \dots, C_{n,k}\}) = x_i$$

and x_i is disclosed to others if $c_k \neq 0$. Therefore, in this case, the participants should conduct additional secure sum calculation before sending the ciphertexts to others.

2) *Randomness and Group Selection*: In fact, in the product calculation protocol, the group \mathbb{G}_1 should be carefully selected to make the input x_i indistinguishable to a random element. We select a cyclic multiplicative group $\mathbb{G}_1 \subset \mathbb{Z}_p$ of prime order q as follows. Find two large prime numbers p, q such that $p = kq + 1$ for some integer k . Then, find a generator h for \mathbb{Z}_p , and set $g_1 := h^{(p-1)/q}$ modulo p (clearly $g_1 \neq 1$ modulo p). Then group \mathbb{G}_1 is generated by g_1 , whose order is q . Here the powers of the numbers in \mathbb{G}_1 belong to an integer group \mathbb{Z}_q .

Next, we show that any input data x_i is computationally indistinguishable to any random element chosen from \mathbb{Z}_p via PRF.

For any i , we have

$$C_i = x_i (g_1^{r_{i+1}} / g_1^{r_{i-1}})^{r_i} = x_i g_1^{(r_{i+1} - r_{i-1}) r_i}.$$

Let x_i be $g_1^{\chi_i}$ and $r_{i+1} - r_{i-1}$ be γ_i , where $\chi_i \in \mathbb{Z}_q$ and $\gamma_i \in \mathbb{Z}_q$ (This is possible since g_1 is a generator of the group \mathbb{G}_1). Then, $C_i = g_1^{\chi_i} g_1^{\gamma_i r_i}$.

Theorem VII.1. $\forall x_i, r_i \in \mathbb{Z}_q, \exists \hat{r}_i, \hat{\chi}_i \in \mathbb{Z}_q$ such that

$$g_1^{\chi_i} g_1^{\gamma_i r_i} = g_1^{\hat{\chi}_i} g_1^{\gamma_i \hat{r}_i} \pmod{p}.$$

Proof: For any $r_i, \hat{r}_i \in \mathbb{Z}_q$, there exists $\hat{\chi}_i \in \mathbb{Z}_q$ such that:

$$\gamma_i (r_i - \hat{r}_i) = \hat{\chi}_i - \chi_i \pmod{q}$$

because q and $(r_i - \hat{r}_i)$ are relatively prime (q is a prime number). Then we have $\hat{\chi}_i \in \mathbb{Z}_q$ for any $r_i \in \mathbb{Z}_q$ such that:

$$g_1^{\gamma_i (r_i - \hat{r}_i)} = g_1^{\hat{\chi}_i - \chi_i} \pmod{p} \Rightarrow g_1^{\chi_i} g_1^{\gamma_i r_i} = g_1^{\hat{\chi}_i} g_1^{\gamma_i \hat{r}_i} \pmod{p}$$

This implies that given the ciphertext C_i , any value x_i is a possible valid data that can produce this 'ciphertext' C_i . ■

According to the Theorem VII.1, we can deduce that χ_i has the same level of randomness as r_i . Therefore, $g_1^{\chi_i}$ is indistinguishable to a random element in \mathbb{G}_1 from other participants' or attackers' perspective, which implies

Theorem VII.2. *The input x_i is computationally indistinguishable to a random element chosen from \mathbb{G}_1 .*

3) *Closure and Group Selection*: We need to guarantee that all the multiplications in the sum protocol are closed in \mathbb{G}_2 . Since $(1 + x_i p) \cdot (g_2^{r_{i+1}} / g_2^{r_{i-1}})^{r_i}$ is the only multiplication throughout the sum protocol, we must carefully choose the group \mathbb{G}_2 such that $1 + x_i p \in \mathbb{G}_2$. We let $\mathbb{G}_2 \subset \mathbb{Z}_{p^2}$ be a cyclic multiplicative group generated by h , which is the generator of \mathbb{Z}_p . Then, the order of \mathbb{G}_2 is $p(p-1)$, and the powers of the numbers in \mathbb{G}_2 belong to an integer group $\mathbb{Z}_{p(p-1)}$. Since $\mathbb{G}_2 = \mathbb{Z}_{p^2} - \{x | x = k \cdot p, \text{ for some integer } k\}$ and $\forall k : 1 + x_i p \neq kp, 1 + x_i p$ belongs to the group \mathbb{G}_2 .

4) *Restriction of the Product and Sum Protocol*: In both protocols, we require that 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 $x_1 x_2$ 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.

C. Complexity

We discuss the computation and communication complexity of the *Advanced Scheme* for each model in this section.

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 II
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)	$O(1)$	$O(p)$

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 III
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:

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

VIII. PERFORMANCE 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 V). 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.

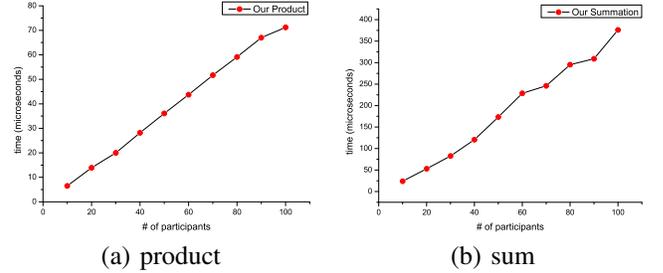


Fig. 3. Running time for product and sum calculation.

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.* [2] (FairplayMP). They implemented the BMR protocol [1], 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) [22][23].

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 IV
RUN TIME (MILLISECONDS) OF FAIRPLAYMP[2]

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 [16]. Lauter *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 V
COMPARISON BETWEEN [16] AND OUR SYSTEM

	Addition	Multiplication
Lauter [16]	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.

IX. CONCLUSION

In this paper, we successfully achieve a privacy-preserving multivariate polynomial evaluation without secure communication channels by introducing our novel secure product and sum calculation protocol. We also show in the discussion that our proposed construction is efficient and secure enough to be applicable in real life. However, our scheme discloses each product part in the polynomial, which gives unnecessary information to attackers. Therefore, our next research will be minimizing the information leakage during the computation

and communication. Another future work is to design privacy preserving data releasing protocols such that certain functions can be evaluated correctly while certain functional privacy can be protected.

REFERENCES

- [1] D. Beaver, S. Micali, and P. Rogaway, “The round complexity of secure protocols,” in *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, 1990, pp. 503–513.
- [2] A. Ben-David, N. Nisan, and B. Pinkas, “Fairplaymp: a system for secure multi-party computation,” in *Proceedings of the 15th ACM conference on Computer and communications security*, 2008, pp. 257–266.
- [3] J. Bethencourt, A. Sahai, and B. Waters, “Ciphertext-policy attribute-based encryption,” in *IEEE Symposium on Security and Privacy*, 2007, pp. 321–334.
- [4] D. Boneh, “The decision diffie-hellman problem,” *Algorithmic Number Theory*, pp. 48–63, 1998.
- [5] C. Castelluccia, A. Chan, E. Mykletun, and G. Tsudik, “Efficient and provably secure aggregation of encrypted data in wireless sensor networks,” *ACM Transactions on 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 *The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, 2005, pp. 109–117.
- [7] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Zhu, “Tools for privacy preserving distributed data mining,” *ACM SIGKDD Explorations Newsletter*, vol. 4, no. 2, pp. 28–34, 2002.
- [8] W. Diffie and M. Hellman, “New directions in cryptography,” *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.
- [9] W. Dong, V. Dave, L. Qiu, and Y. Zhang, “Secure friend discovery in mobile social networks,” in *IEEE INFOCOM*, 2011, pp. 1647–1655.
- [10] T. ElGamal, “A public key cryptosystem and a signature scheme based on discrete logarithms,” in *Advances in Cryptology*, Springer, 1985, pp. 10–18.
- [11] C. Gentry, “Fully homomorphic encryption using ideal lattices,” in *Proceedings of the 41st annual ACM symposium on Theory of computing*, 2009, pp. 169–178.
- [12] C. Gentry and S. Halevi, “Implementing gentry’s fully-homomorphic encryption scheme,” *Advances in Cryptology—EUROCRYPT 2011*, pp. 129–148, 2011.
- [13] O. Goldreich, “Secure multi-party computation,” *Manuscript. Preliminary version*, 1998.
- [14] W. He, X. Liu, H. Nguyen, K. Nahrstedt, and T. Abdelzaher, “Pda: Privacy-preserving data aggregation in wireless sensor networks,” in *IEEE INFOCOM*, 2007, pp. 2045–2053.
- [15] T. Jung, X. Li, Z. Wan, and M. Wan, “Privacy preserving cloud data access with multi-authorities,” in *IEEE INFOCOM*, 2013.
- [16] K. Lauter, M. Naehrig, and V. Vaikuntanathan, “Can homomorphic encryption be practical,” *Preprint*, 2011.
- [17] X. Li and T. Jung, “Search me if you can: privacy-preserving location query service,” in *IEEE INFOCOM*, 2013.
- [18] A. Menezes, P. Van Oorschot, and S. Vanstone, *Handbook of applied cryptography*, CRC, 1997.
- [19] R. Sheikh, B. Kumar, and D. Mishra, “Privacy preserving k secure sum protocol,” *Arxiv preprint arXiv:0912.0956*, 2009.
- [20] E. Shi, T. Chan, E. Rieffel, R. Chow, and D. Song, “Privacy-preserving aggregation of time-series data,” in *Proceedings of NDSS*, vol. 17, 2011.
- [21] V. Verykios, E. Bertino, I. Fovino, L. Provenza, Y. Saygin, and Y. Theodoridis, “State-of-the-art in privacy preserving data mining,” *ACM Sigmod Record*, vol. 33, no. 1, pp. 50–57, 2004.
- [22] A. Yao, “Protocols for secure computations,” in *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, 1982, pp. 160–164.
- [23] —, “How to generate and exchange secrets,” in *27th Annual Symposium on Foundations of Computer Science*, 1986, pp. 162–167.
- [24] L. Zhang, X. Li, Y. Liu, and T. Jung, “Verifiable private multi-party computation: ranging and ranking,” in *IEEE INFOCOM Mini-Conference*, 2013.