Secure Function Evaluation Based on Secret Sharing and Homomorphic Encryption

Shantanu Rane, Wei Sun, Anthony Vetro

TR2009-084   December 2009

Abstract

Consider the following problem in secure multiparty computation: Alice and Bob posses integers X and y respectively. Charlie is a researcher who would like to compute the value of some function f(x,y). The requirement is that Charlie should not gain any knowledge about x and y other than that which can be obtained from the function itself. Moreover, Alice and Bob do not trust each other and should not gain knowledge about each other's data. This paper contains initial work on a methodology to enable such secure function evaluation using additive and multiplicative homomorphisms as cryptographic primitives instead of oblivious transfer. It is shown that Charlie can compute the encrypted value of any polynomial in x and y. We present two secure function evaluation protocols for semi-honest participants that can be extended to polynomial functions of an arbitrary number of variables.

Allerton Conference 2009

Secure Function Evaluation based on Secret Sharing and Homomorphic Encryption

Shantanu Rane, Wei Sun and Anthony Vetro
Mitsubishi Electric Research Laboratories
Cambridge, MA 02139
{rane,weisun,avetro}@merl.com

***Abstract*—Consider the following problem in secure multiparty computation: Alice and Bob possess integers x and y respectively. Charlie is a researcher who would like to compute the value of some function $f(x, y)$. The requirement is that Charlie should not gain any knowledge about x and y other than that which can be obtained from the function itself. Moreover, Alice and Bob do not trust each other and should not gain knowledge about each other's data. This paper contains initial work on a methodology to enable such secure function evaluation using additive and multiplicative homomorphisms as cryptographic primitives instead of oblivious transfer. It is shown that Charlie can compute the encrypted value of any polynomial in x and y. We present two secure function evaluation protocols for semi-honest participants that can be extended to polynomial functions of an arbitrary number of variables.**

I. Introduction

This paper is concerned with privacy-preserving data analysis. The data in question is assumed to be distributed among several mutually untrusting data centers. It is required to analyze this data to obtain some statistical information or to form some inference by evaluating a function of the data. To preserve the privacy of the data centers, no information about the data should be divulged beyond the result of the analysis. One example of such a situation arising in practice is as follows: An external agency, henceforth labelled the "researcher," is entrusted with the task of gathering statistical information about disease symptoms observed in a number of hospitals. With today's amplified concerns over the privacy of patients, each hospital must ensure that this analysis can be facilitated without divulging its patients' private information. Performing computation in this secure fashion presents many challenges: First, is it possible to obtain non-trivial analytical results under these strict privacy requirements? In other words, what class of functions can the researcher hope to evaluate while respecting the privacy of the untrusting data centers? Second, what is the transmission cost and computational overhead incurred by the data centers and the researcher? Third, how does the overhead scale with the number of participating data centers? Fourth, is such a protocol resistant to collusion between the data centers? This paper presents initial work on the use of secret sharing and homomorphic mappings to construct practical secure function evaluation protocols that address some of these questions.

This problem is posed under the umbrella of secure multiparty computation, in which several parties execute a protocol to securely evaluate a function of several variables. The function to be evaluated is usually modeled as a combinatorial circuit [1], [2], or as a polynomial over a finite field [3], [4]. In the most general settings considered for secure multiparty computation [1], [5], each participant communicates with a large number of participants (possibly all other participants) such that, at the end of the protocol, each party knows the value of the function while keeping its input private from all other parties. Much work has been devoted to securing these protocols against different classes of adversaries: passive colluders, Byzantine colluders, coercive colluders [6]. It has been shown that any function can be evaluated securely given a cryptographic primitive known as Oblivious Transfer (OT). OT itself is implemented using assumptions on the computational intractability of certain problems, such as factoring large integers, noisy polynomial reconstruction, the RSA problem, the quadratic residuosity problem to name a few [7]. For general secure multiparty computation with densely connected participants, the protocols are extremely intensive in terms of computational and communication overhead, even for relatively simple functions. For instance, securely evaluating a function of two variables owned by separate parties requires a protocol which implements one OT per input wire of the resulting combinatorial circuit, and performs one encryption per bit of the input [8].

In our work, we consider a particular realization of secure multiparty computation in which the data centers can support only a limited number of communication links, while the researcher and possibly some assisting servers can provide a communication link with each data center. In general, such asymmetric realizations are much more susceptible to privacy breaches caused by a well-chosen set of colluders. Hence, it becomes necessary to analyze the collusion resistance of protocols based on these realizations. In the first protocol presented in this paper, the individual data centers do nothing other than providing the data for computation; the burden of secure computation is shouldered entirely by the researcher and one or more untrusted assisting servers. In the second protocol, which is more resilient to collusion attacks, the data centers are still limited to one communication link, but they perform encryptions or decryptions as directed by the researcher.

To begin with, the problem is framed as one of secure function evaluation with three untrusting parties: two data centers (Alice and Bob) and a researcher (Charlie), all of whom are assumed to be semi-honest. It is shown in Section II that, using additive secret sharing and additive homomorphic mappings, it is possible to securely compute any polynomial in two variables. In this simplified situation, one of the data centers (Alice) also serves as an assisting server. In Section III, the protocol for polynomial evaluation is extended to an arbitrary number of semi-honest data centers and hence, to an arbitrary number of variables. It is shown that this protocol resists collusion between the researcher and the data centers, but fails in the case of collusion between the researcher and the assisting servers. To address this, an alternative protocol is presented in which the assisting server is removed, giving a star topology with the untrusted researcher directing communications to and from the data centers. Beyond polynomial evaluation, this protocol can also be used when the functions are not polynomials to begin with, but the problem can be still converted into one of polynomial evaluation. Two examples of this kind, viz., comparison of two integers and determination of the error pattern between two binary strings, are considered in Section IV.

II. Secure Function Evaluation

Let $\xi(\cdot)$ be an additively homomorphic mapping, i.e., $\xi(m_1 + m_2) = \xi(m_1)\xi(m_2)$ and $\xi(km_1) = \xi(m_1)^k$ for integer messages m_1, m_2 and a constant integer k. Examples of such a mapping include the Paillier

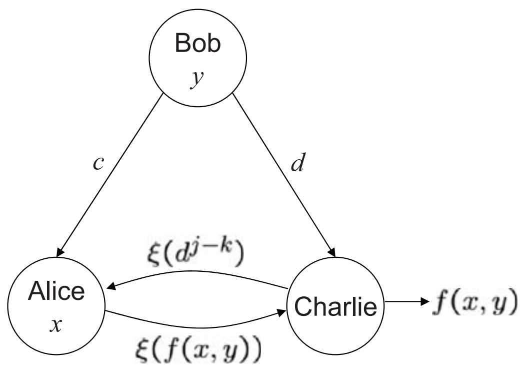


Fig. 1. A three party protocol for privacy-preserving function evaluation: Charlie computes a polynomial function of Alice and Bob's inputs without knowing the inputs themselves.

cryptosystem [9] and the Benaloh cryptosystem[10]. To simplify notation, the encryption and decryption keys are not explicitly specified; the encryption and decryption functions are simply referred to as $\xi(\cdot)$ and $\xi^{-1}(\cdot)$ and the ownership of the relevant keys is assumed to be clear from the context. Let $f(x,y) = \sum_{i,j\geq 0} \gamma_{i,j} x^i y^j$ be a polynomial with integer variables x and y and integers coefficients $\gamma_{i,j}$. Suppose that Alice and Bob possess x and y respectively and want to keep their data private. Charlie, the researcher, wants to compute $f(x,y)$ without explicitly finding out x and y.

By the additively homomorphic property of $\xi(\cdot)$, we have

$$\xi(f(x,y)) = \prod_{i,j\geq 0} \xi(\gamma_{i,j} x^i y^j) = \prod_{i,j\geq 0} \xi(x^i y^j)^{\gamma_{i,j}}. \quad (1)$$

To show the feasibility of computing $\xi(f(x,y))$, it suffices to show the feasibility of computing $\xi(x^i y^j)$, i.e., the encryptions of the individual monomials. Let $y = c + d$ where c and d are integers. Then

$$\begin{aligned}\xi(x^i y^j) &= \xi(x^i (c+d)^j) = \xi(x^i \sum_{k=0}^{j} \binom{j}{k} c^k d^{j-k}) \\ &= \xi(\sum_{k=0}^{j} \binom{j}{k} x^i c^k d^{j-k}) = \prod_{k=0}^{j} \xi(x^i c^k d^{j-k})^{\binom{j}{k}}\end{aligned} \quad (2)$$

A. Protocol for Secure Computation

Now, consider the following protocol (Fig. 1) for secure computation of $\xi(f(x,y))$ using the additive homomorphic property and additive secret sharing. For the time being, assume that Alice, Bob and Charlie are all honest but curious, i.e., each of them will follow the steps of the protocol but can store the intermediate values at each step in which they are involved, and thus try to obtain information about the other parties' inputs. Consider also that the homomorphic mappings are semantically secure, thus repeated encryptions of the same plaintext result in different ciphertexts. Both the Paillier and Benaloh cryptosystems achieve semantic security via a random parameter that is used during encryption but not during decryption.

1) Charlie generates a key pair for homomorphic encryption, consisting of a public encryption key and a private decryption key.
2) Bob splits y into additive shares c and d, i.e., $y = c + d$. He sends c to Alice and d to Charlie.
3) Charlie encrypts d^{j-k} for all $k = 0, 1, ..., j$ and transmits $\xi(d^{j-k})$ to Alice.
4) Using the additive homomorphism of $\xi(\cdot)$, Alice computes $\xi(d^{j-k})^{(x^i c^k)} = \xi(x^i c^k d^{j-k})$ for each $k = 0, 1, ..., j$, and then computes $\xi(x^i y^j)$ according to (2). Then she obtains $\xi(f(x, y))$ using (1) and sends it to Charlie.
5) Charlie decrypts $f(x, y)$

B. Security Analysis

1) Alice knows only the plaintext share c of Bob's data y. All the other operations that she performs are in the homomorphically encrypted domain. Thus, she finds out nothing about y.
2) Bob does not receive any information either from Alice or Charlie.
3) Charlie knows only the plaintext share d of Bob's y. In the last step of the protocol, he recovers $f(x, y)$. If $f(x, y)$ is a polynomial in a single variable, i.e., if $x = 1$ or $y = 1$ in the above protocol, then Charlie can obtain candidate values for that variable via polynomial factorization. In all other cases, it is computationally infeasible for Charlie to obtain x or y.

C. Computational and Transmission Overhead

Suppose that the polynomial $f(x, y)$ has degree M in the variable x, degree N in the variable y. The communication and computational load incurred by Alice, Bob and Charlie are approximated below. While computing the number of multiplications, only those conducted in the encrypted domain are counted as they are much more intensive than plaintext multiplications. Bob conducts only two one-way communications, one with Alice and one with Charlie in Step 2. Charlie performs $O(N)$ encryptions in Step 3 and one decryption in Step 5. Alice performs no encryptions or decryptions, but incurs one communication with Charlie at the end of Step 4. Alice's resources are almost entirely used up in computation in Step 4. She performs $O(S^M)$ multiplications in the encrypted domain, where S is the size of the field from which x and y are chosen. The computational and transmission overhead of the three parties is summarized in Table I.

	# Encryptions or Decryptions	# Transmissions	# Multiplications
Alice	-	1	$O(S^M)$
Bob	-	2	-
Charlie	$O(N)$	1	-

TABLE I
RESOURCES USED BY ALICE, BOB AND CHARLIE FOR THE CASE OF SEMI-HONEST PARTICIPANTS.

III. EXTENSIONS

A. Numerous Participating Data Centers

Table I indicates that Bob's computational load and communication overhead are both very small. Meanwhile, Alice shoulders the computation overhead, while Charlie performs the tasks of key generation, encryption and decryption. This asymmetric distribution of responsibilities permits scaling the protocol to numerous participating data centers which do not possess large computational resources. In this extended setting, there is a single researcher, Charlie, who wants to compute a function of several variables $f(x_1, x_2, x_3, ...)$. There is a single assisting server, Alice, who provides computing resources. Finally, there are many data centers $P_1, P_2, P_3, ...$, who allow Charlie to evaluate the function f on the condition that he will not gain access to their individual data. This configuration is shown in Figure 2. Assuming honest but curious parties, the protocol also ensures, via additive secret sharing, that the data of the data centers is secure from each other and from Alice. As an example, suppose that there are two data centers P_1 and P_2 who split their data as $x_1 = c_1 + d_1$ and

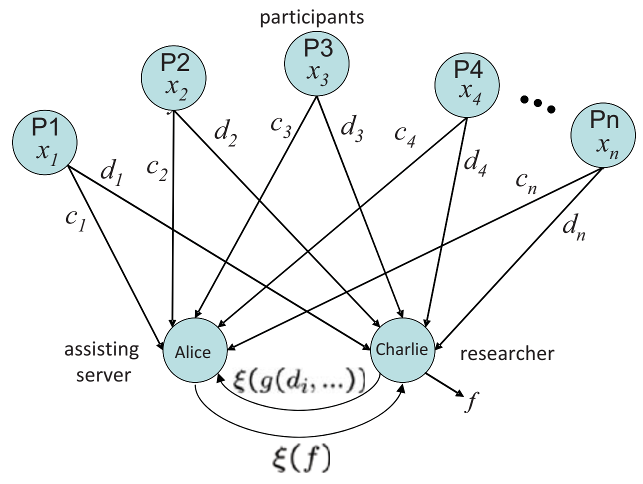


Fig. 2. With the help of an assisting server, the researcher can securely evaluate the value of a function with n arguments without knowing the values of the function's arguments.

$x_2 = c_2 + d_2$ respectively. Then,

$$\begin{aligned}\xi(x_1^i x_2^j) &= \xi((c_1 + d_1)^i (c_2 + d_2)^j) \\ &= \xi(\sum_{k=0}^{i} \binom{i}{k} c_1^k d_1^{i-k} \sum_{k'=0}^{j} \binom{j}{k'} c_2^{k'} d_2^{j-k'}) \\ &= \prod_{k=0}^{i} \prod_{k'=0}^{j} \xi(\binom{i}{k}\binom{j}{k'} c_1^k d_1^{i-k} c_2^{k'} d_2^{j-k'}) \\ &= \prod_{k=0}^{i} \prod_{k'=0}^{j} \xi(d_1^{i-k} d_2^{j-k'})^{\binom{i}{k}\binom{j}{k'} c_1^k c_2^{k'}} \qquad (3)\end{aligned}$$

Comparing with Section II, P_1 and P_2 perform the role of Bob and incur low communication and computation overhead. Alice and Charlie can execute the protocol as before. Note that, in the example chosen here, Alice is only supplying computational resources and not contributing any data to the function evaluation. Provided that Alice has enough computational resources to perform the increased number of multiplications, the protocol can be readily scaled to accommodate an arbitrary number of data centers.

B. Collusion Attacks

Up to this point, all parties in the protocol have been assumed to be honest but curious. We now analyze the different ways in which one or more parties may collude in an effort to subvert the privacy requirements and discover the inputs owned by the data centers. Of course, the set of colluders might want to interfere with the protocol with a related aim - to provide the researcher with an erroneous value of the function f. In the sequel, we regard the privacy of the data centers P_i as paramount, and therefore only analyze privacy attacks in this paper. In this analysis, there are n data centers $P_1, P_2, ..., P_n$, an assisting server, Alice, and a researcher, Charlie.

1) Suppose that the value of the function f is made public by Charlie at the end of the protocol. In this case, a set of colluders (the coalition) containing $t \leq n-2$ data centers cannot discover anything about the data of the remaining $n-t$ data centers. If the set of colluders consists of $n-1$ data centers, then f reduces to a polynomial in one variable. The coalition can find the roots of this polynomial, one of which is the input of the remaining data center.
2) Suppose that the value of the function f is not made public, and only retained by Charlie. In this case, a coalition with $t \leq n-1$ data centers cannot discover anything about the data of the remaining $n-t$ data centers.
3) Suppose that the coalition consists of the assisting server, Alice, as well as any t data centers. Since Alice lacks the decryption key for the homomorphic mapping, it is computationally intractable for the coalition to learn the data of the remaining $n-t$ data centers. As above, when the value of f is made public, a coalition containing Alice as well as $t \leq n-2$ colluders will be unsuccessful in discovering the data of the remaining $n-t$ data centers.
4) Suppose that the coalition contains the researcher, Charlie, as well as any t data centers. Again, in this case, a coalition of size $t \leq n-2$ cannot obtain the data of the remaining $n-t$ data centers.
5) Suppose the coalition consists of the researcher and the assisting server. Owing to the asymmetric distribution of responsibilities in our protocol, this is the most serious kind of collusion attack and therefore the weakest link that an adversary can exploit. With the protocol as described in Section II, this attack would enable the coalition to obtain the data of all n data centers. This type of attack reduces the network to a star topology with the n data centers interacting with a centralized researcher who has direct access to every protocol transmission. The next subsection presents a collusion-resistant protocol for computing polynomials for a star topology. We note that the collusion attack described above could be mitigated using other techniques, such as by

including a large number of assisting servers to share secrets, or by allowing a limited amount of direct communication among the data centers themselves. However, owing to our assumption that the data centers are constrained in the number of communication lines they can support, we do not consider such strategies.

C. Robustness to Collusion Attacks

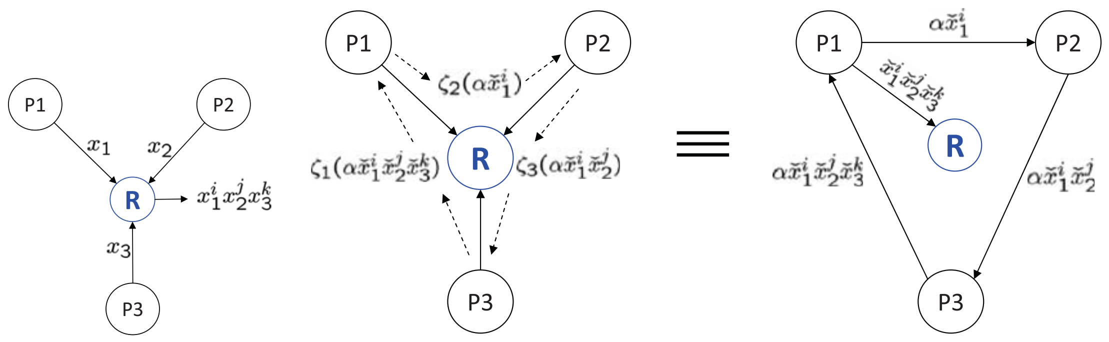
When the adversary corrupts both the researcher and the assisting server, he has access to both the additive shares c_i and d_i and can trivially obtain the participants' data $x_i = c_i + d_i$. In other words, the topology of Fig. 2 reduces to a star topology. This is shown in Fig. 3 for the case of 3 participating data centers. The researcher and the assisting server have been collapsed into a single entity. In addition to trivially discovering every participant's data, another obvious consequence of this attack is that the adversary gains access to each individual monomial in the expression of the function f. Note, from Section II, the monomials were evaluated in the encrypted domain by the assisting server. These were combined using additive homomorphism and the result was transmitted to the researcher, who proceeded to decrypt the polynomial f. When an adversary corrupts both the researcher and the assisting server, then he also obtains access to the unencrypted monomials.

In a star topology with a centralized researcher, the problem of data privacy at the researcher would be solved if he only receives encrypted transmissions from the data centers. Correctness of the protocol would be ensured if this encryption preserves the algebraic structure, namely allows computation of encrypted versions of the constituent monomials. Based on this realization, we present below a protocol to compute the constituent monomials. Let $\theta(\cdot)$ be a multiplicatively homomorphic mapping, i.e., $\theta(m_1 \cdot m_2) = \theta(m_1) \cdot \theta(m_2)$ and $\theta(m_1^k) = (\theta(m_1))^k$ for integer messages m_1, m_2 and a constant integer k. The El Gamal cryptosystem [11] is an example of a semantically secure multiplicative homomorphic cryptosystem.

To facilitate explanation, we assume that the monomial is $x_1^i x_2^j x_3^k$. Now the protocol for securely evaluating this monomial is as follows:

1) Suppose that the data centers P_m hold integer inputs x_m. Let each data center have access to a unique encryption/decryption key pair for a public key cryptosystem. The encryption key of every participant m is public and the encryption function is denoted by $\zeta_m(\cdot)$. The decryption key of every data center is privately held throughout the protocol.
2) The researcher generates an encryption/decryption key for multiplicative homomorphic encryption. The encryption key is publicly available to all data centers, and the encryption of x_m is denoted by $\theta(x_m) = \check{x}_m$. Thus, by the multiplicative homomorphic property, $\theta(x_m^i) = \check{x}_m^i$. The decryption key is privately held by the researcher throughout the protocol.
3) Participant P_1 encrypts x_1^i using the researcher's public encryption key and obtains $\theta(x_1^i) = (\theta(x_1))^i = \check{x}_1^i$. He then chooses a non-zero integer α at random and obtains $\zeta_2(\alpha\check{x}_1^i)$ using the public encryption key of P_2. He transmits this result to the researcher, who forwards it to P_2.
4) P_2 obtains $\alpha\check{x}_1^i$ by decryption. He computes $\check{x}_2^j$ as above, using the researcher's public encryption key. Then, he computes $\zeta_3(\alpha\check{x}_1^i\check{x}_2^j)$ using the public encryption key of P_3. He transmits it to the researcher, who forwards it to P_3.
5) P_3 obtains $\alpha\check{x}_1^i\check{x}_2^j$ by decryption, computes $\zeta_1(\alpha\check{x}_1^i\check{x}_2^j\check{x}_3^k)$ using the public encryption key of P_1. He transmits it to the researcher who forwards it to P_1.
6) P_1 obtains $\alpha\check{x}_1^i\check{x}_2^j\check{x}_3^k$, removes α by division and forwards $\check{x}_1^i\check{x}_2^j\check{x}_3^k = \theta(x_1^i x_2^j x_3^k)$ to the researcher.
7) The researcher uses his private decryption key for $\theta(\cdot)$ to obtain $x_1^i x_2^j x_3^k$.

In this simple example, P_2 cannot discover x_1 because he does not know α or the decryption key for $\theta(\cdot)$. Similarly, P_3 cannot discover x_1 and x_2 and so on. The researcher has access to all the encrypted transmissions from the data centers but possesses no keys to decrypt them. In the semi-honest case with no collusions, this protocol is secure for any monomial that contains 2 or more variables. In general, for a monomial containing $s > 2$ variables, a collusion consisting of $t \leq s - 2$ data centers will be unsuccessful in compromising the the data of the remaining $s - t$ data centers. However, if the researcher joins the colluders, a coalition of $s/2$ properly chosen data centers can discover the data of the remaining $s/2$ data centers. This result can be verified, for example, by considering a collusion attack in which the researcher colludes with all the even numbered data centers. Still, this protocol is more resilient to loss of data privacy in the presence of collusion attacks, compared to the one in Section II. Even if any subset of the data centers collude (unknown to the researcher), they


P1
P2
P3
R
x_1
x_2
x_3
$x_1^i x_2^j x_3^k$
$\zeta_2(\alpha\breve{x}_1^i)$
$\zeta_1(\alpha\breve{x}_1^i\breve{x}_2^j\breve{x}_3^k)$
$\zeta_3(\alpha\breve{x}_1^i\breve{x}_2^j)$
$\equiv$
$\alpha\breve{x}_1^i$
$\breve{x}_1^i\breve{x}_2^j\breve{x}_3^k$
$\alpha\breve{x}_1^i\breve{x}_2^j\breve{x}_3^k$
$\alpha\breve{x}_1^i\breve{x}_2^j$

(a) Loss of Privacy

(b) Collusion Resistant Protocol

Fig. 3. (a) Attacking the researcher as well as the assisting server in Fig. 2 converts the secret sharing topology into a star topology in which the adversary trivially obtains the contributions of the data centers as well as the individual monomials in f. (b) A modification in which the participants use multiplicative homomorphic encryption to resist catastrophic loss of privacy from collusion attacks mounted by the researcher or by a subset of data centers.

cannot compromise the privacy of the remaining data centers. For example, if P_1 and P_3 collude, they can only discover $\breve{x}_2^j = \theta(x_2^j)$ but cannot obtain the value of x_2.

The price paid for improved privacy preservation is increased computation at the data centers. The researcher performs only one decryption at the end of the protocol, and his role is restricted to exchanging public keys, notifying the participants about the monomial to be evaluated and directing the transmissions to the appropriate participants. The reader will note that, in the absence of the researcher, the connections between the participants form a ring $P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_1$ between the participants who own the relevant variables. If each data center is connected to every other data center, then, a ring can be set up between data centers corresponding to any monomial. However, if a densely connected network is not available, this ring topology can be expensive to implement for arbitrary monomials. Instead, with a star connection, every data center makes only one connection to the researcher who sets up the appropriate ring required to compute the monomial, and hence the polynomial function.

The protocol just described also works if the monomial products such as $x_1^i x_2^j x_3^k$ are replaced by sums of the form $x_1^i + x_2^j + x_3^k$. This modification allows the computation of statistical properties such as the mean, variance and higher order moments of the data stored in the data centers. If, as explained above, we consider the star network as merely facilitating the construction of arbitrary rings of data centers, then this method of computing sums of powers becomes similar to the scheme described in [12]. In that scheme, the first data center in the ring introduces a constant α and removes it from the final result. Our scheme is similar in that respect, however the introduction of a semi-honest researcher at the center of the star topology necessitates the use of encryption by the data centers. For the application of finding secure summations, Karr *et al.* [12] also note that it is advantageous to convert the ring to a star topology with a centralized server directing the traffic to appropriate data centers. Additionally, we have introduced a second encryption in the protocol, namely the multiplicatively homomorphic encryption using the researcher's public key, which prevents participants from colluding to obtain the data of the other participants, while still ensuring that the monomial can be evaluated by the researcher. We note that by combining secure summations and secure products, the protocol can be used to evaluate many functions that are useful in practice. A few of these are explained in the next section.

IV. Evaluation of Some Useful Functions

The above protocol allows the researcher to securely determine some practically useful functions on the data of any subset of the n participants. Some examples of these include the Hamming distance between binary sequences, the error pattern between two binary sequences, maximum of two or more integers,. i.e., the secure millionaires problem in secure multiparty computation [1], squared distance between integer sequences, averages and higher moments of two or more integer sequences. Some of these will be described in following subsections. There has been previous work on these problems, notably secure comparison [13] and secure auctions [14]. In these works, participants interact with one or more assisting servers to find out who possesses the larger of two numbers. Since two protocols have already been described in Section II and III-C, we restrict the treatment in this section to describing the application and obtaining an expression for the function f. In some of these examples, f is obvious from the application, while in some others, this may not be the case.

A. Secure Comparison of Two Very Large Integers

Suppose that Alice and Bob possess two very large positive integers x and y, and the researcher is interested in knowing which is the larger of the two, without finding out the integers themselves. This is an instance of the secure millionaires problem. The researcher can trivially construct $f(x, y) = x - y$ and determine the larger of x and y from the sign of $f(x, y)$. However, this would require secret sharing of x and y which are assumed to be very large. An efficient solution can be obtained by converting the large numbers into binary sequences $x = x_1x_2...x_m$ and $y = y_1y_2...y_m$, where m is the most significant bit. For each $i = 1, 2, ..., m$, define

$$\begin{aligned} f_i &= x_i - y_i + \sum_{j=i+1}^{m} x_j \oplus y_j \\ &= x_i - y_i + \sum_{j=i+1}^{m} x_j + y_j - 2x_jy_j \qquad (4) \end{aligned}$$

which is a polynomial in $x_i, x_{i+1}, ..., x_m, y_i, y_{i+1}, ..., y_m$. Thus, Alice acting as the owner of $x_i, x_{i+1}, ..., x_m$, Bob acting as the owner of $y_i, y_{i+1}, ..., y_m$, can interact with a researcher who evaluates f_i. Note here, that since the x_i and y_i are bits, additive secret sharing of the form $x_i = c_1 + d_1$ and $y_i = c_2 + d_2$ can be performed with much smaller values of the individual shares, depending upon the privacy desired. This is in contrast to the very large individual shares that would be necessary if the comparison was done directly on the integers x and y, instead of the individual bits. There is another advantage of this method which will become clear in the next paragraph. Before that, consider the following result which justifies calculating f_i in (4) to compare the integers x and y.

Proposition 1: $x < y$ if and only if there exists an $i \in \{m, m-1, ..., 2, 1\}$ such that $f_i = -1$.

Proof: It is easy to verify that $f_i = -1$ if and only if $x_i - y_i + 1 = 0$ and $\sum_{j=i+1}^{m} x_j \oplus y_j = 0$ since both $x_i - y_i + 1$ and $\sum_{j=i+1}^{m} x_j \oplus y_j$ are nonnegative integers. But, $x_i - y_i + 1 = 0$ implies $(x_i, y_i) = (0, 1)$ and $\sum_{j=i+1}^{m} x_j \oplus y_j = 0$ implies $x_j = y_j$ for all $j > i$. These two statements together give $x < y$.

The above proposition provides a way to compare x_i and y_i. Starting from the most significant bits, the researcher calculates $f_m, f_{m-1}, ...$ until he finds an i such that $f_i = -1$. As soon as such an i is found, then the comparison concludes with the result that $x < y$. The remaining least significant bits in positions $1, 2, ..., i-1$ need not be tested at all. This is another reason why this method of operating on the individual bits of x and y is not as wasteful as it initially appears. If the researcher finds that there is no i satisfying $f_i = -1$, he concludes that $x \geq y$. A similar approach to comparing two numbers is used in a three party protocol in [13] which is slightly different from the scenario considered here. In that work, Alice owns x and interacts with Bob, who owns y, and an untrusted assisting server to determine which of their numbers is larger. In our setting, the researcher does not own either x or y but is interested in the result. Via several pairwise comparisons, this protocol allows the researcher to determine the largest of n integers, to rank order them, and so on.

B. Error Pattern Between Binary Sequences

Suppose that Alice and Bob have massive binary sequences $\{x\} = x_1x_2...x_m$ and $\{y\} = y_1y_2...y_m$ respectively, and the researcher wants to know only the places in which the bits of $\{x\}$ and $\{y\}$ differ. He defines the polynomials $f_i = x_i \oplus y_i = x_i + y_i - 2x_iy_i$ for $i = 1, 2, 3, ..., m$. and computes f_i to obtain the error pattern between $\{x\}$ and $\{y\}$.

C. Statistical Measures and Distortions

As noted in Section III-C, replacing the products of powers of x_i in the protocol by sums of powers of x_i allows the researcher to compute statistical measures such as averages, variances and higher moments on the data owned by the data centers. Further, the protocols described in this paper allow privacy-preserving computation of distortions such as the mean squared error between images owned by two data centers.

V. Conclusion

This paper presented two protocols for privacy preserving data analysis, specifically the case in which a researcher evaluates functions in many variables owned by mutually untrusting data centers. The protocols are designed from the point of view of assigning low communication overhead to the data centers, while the major computation load is handled by the researcher and one or more assisting servers. In the first protocol, any polynomial function can be evaluated by using simple additive secret sharing at the data centers and additive homomorphic functions at the researcher. From the point of view of data privacy, the most damaging attack on this protocol would be for an adversary to simultaneously corrupt the researcher as well as the assisting servers; this would compromise the data of all data centers who are connected to that assisting server. This problem is addressed in the second protocol using multiplicative secret sharing to enable secure computation of monomials in a star topology where the data centers interact with a centralized researcher. The protocols can be been utilized to evaluate several useful functions, including finding the error pattern between binary streams, comparing large integers without revealing them to the researcher, and computation of statistical moments of the inputs contributed by the data centers.

REFERENCES

[1] Andrew Chi-Chih Yao, "How to Generate and Exchange Secrets," in *Proceedings of the 27th Annual Symposium on Foundations of Computer Science (FOCS)*, Washington, DC, USA, 1986, pp. 162–167, IEEE Computer Society.

[2] Andrew Chi-Chih Yao, "Protocols for Secure Computations (Extended Abstract)," in *Proceedings of the 23th Annual Symposium on Foundations of Computer Science (FOCS)*. 1982, pp. 160–164, IEEE Computer Society.

[3] M. Naor and B. Pinkas, "Oblivious polynomial evaluation," *SIAM Journal on Computing*, vol. 35, no. 5, pp. 1254–1281, 2006.

[4] Y. Lindell and B. Pinkas, "Privacy Preserving Data Mining," in *CRYPTO '00: Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology*, London, UK, 2000, pp. 36–54, Springer-Verlag.

[5] O. Goldreich, S. Micali, and A. Widgerson, "How to Play Any Mental Game," in *Proceedings of the 19th ACM Symposium on the Theory of Computing*, New York, NY, May. 1987, pp. 218–229.

[6] S. Goldwasser, "Multiparty Computations: Past and Present," in *Proceedings of the 16th ACM Symposium on Principles of Distributed Computing*, Santa Barbara, CA, Aug. 1997, pp. 1–6.

[7] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 2001.

[8] B. Pinkas, "Cryptographic Techniques for Privacy-Preserving Data Mining," *SIGKDD Explorations, the newsletter of the ACM Special Interest Group on Knowledge Discovery and Data Mining*, Jan. 2003.

[9] Pascal Paillier, "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes," in *Advances in Cryptology, EUROCRYPT 99*. 1999, vol. 1592, pp. 233–238, Springer-Verlag, Lecture Notes in Computer Science.

[10] J. Benaloh, "Dense Probabilistic Encryption," in *Proceedings of the Workshop on Selected Areas of Cryptography*, Kingston, ON, Canada, May 1994, pp. 120–128.

[11] T. El Gamal, "A Public Key Cryptosystem and A Signature Scheme Based on Discrete Logarithms," *IEEE Transactions on Information Theory*, vol. 4, pp. 469–472, Jul. 1985.

[12] A. Karr, W. Fulp, F. Vera, S. Young, X. Lin, and J. Reiter, "Secure Privacy-Preserving Analysis of Distributed Databases," *American Statistical Association, Technometrics, Journal*, , no. 3, pp. 335–345, Aug 2007.

[13] I. Damgård, M. Geisler, and M. Krøigård, "Homomorphic encryption and secure comparison," *International Journal of Applied Cryptography*, vol. 1, no. 1, pp. 22–31, Jan. 2008.

[14] M. Jakobsson and A. Juels, "Mix and match: Secure function evaluation via ciphertexts," in *Advances in Cryptology–ASIACRYPT'00, 6th International Conference on the Theory and Application of Cryptology and Information Security*, Kyoto, Japan, Dec. 2000, pp. 162–177.