A Practical Approach to Solve Secure Multi-party Computation Problems·

Wenliang Du and Zhijun Zhan
Center for Systems Assurance
Department of Electrical Engineering and Computer Science
Syracuse University, Syracuse, NY 13244
Email: wedu, zhzhan@ecs.syr.edu

ABSTRACT

Secure Multi-party Computation (SMC) problems deal with the following situation: Two (or many) parties want to jointly perform a computation. Each party needs to contribute its private input to this computation, but no party should disclose its private inputs to the other parties, or to any third party. With the proliferation of the Internet, SMC problems becomes more and more important. So far no practical solution has emerged, largely because SMC studies have been focusing on zero information disclosure, an *ideal* security model that is expensive to achieve.

Aiming at developing practical solutions to SMC problems, we propose a new paradigm, in which we use an *acceptable* security model that allows partial information disclosure. Our conjecture is that by lowering the restriction on the security, we can achieve a much better performance. The paradigm is motivated by the observation that in practice people do accept a less secure but much more efficient solution because sometimes disclosing information about their private data to certain degree is a risk that many people would rather take if the performance gain is so significant. Moreover, in our paradigm, the security is adjustable, such that users can adjust the level of security based on their definition of the acceptable security. We have developed a number of techniques under this new paradigm, and are currently conducting extensive studies based on this new paradigm.

Keywords
Privacy, security, secure multi-party computation.

1. INTRODUCTION

The proliferation of the Internet has triggered tremendous opportunities for cooperative computation, where people are cooperating with each other to conduct computation tasks based on the inputs they each supply. These computations could occur between trusted partners, between partially trusted partners, or even between competitors. For example, two competing financial organizations might jointly invest in a project that must satisfy both organizations' private and valuable constraints, two countries might want to plan a joint military action but each country has some secret information that cannot be shared. Usually, to conduct these computations, one must know inputs from all the participants, but if nobody can be trusted enough to know all the inputs, privacy becomes a critical issue.

A more general form of the problem is described in the following: two or more parties want to conduct a computation based on their private inputs, but neither party is willing to disclose its own input to anybody else. The problem is how to conduct such a computation while preserving the privacy of the inputs. This problem is referred to as Secure Multi-party Computation problem (SMC) in the literature [25]. Generally speaking, A secure multi-party computation (SMC) problem deals with computing any function on any input, in a distributed network where each participant holds one of the inputs, ensuring that no more information is revealed to a participant in the computation than can be computed from that participant's input and output [20].

The greatest challenge that all the solutions need to face is how to satisfy the security requirement. To formally define security [18], traditional SMC studies introduce the concepts of the *ideal* model and the *real* model. In the ideal model, the real parties are joined by a (third) trusted party, and the computation is performed via this trusted party. In the real model, the real SMC protocol is executed (and there exist no trusted third parties). A protocol in the real model is said to be *secure with respect to certain adversarial behavior* if the possible real executions with such an adversary can be "simulated" in the ideal model. Loosely speaking, whatever information disclosure that can occur in the real model could also occur in the ideal model. The security model based on the above definition of security will be called the *ideal security model* throughout this paper (to avoid the confusion of the ideal security model with the ideal model, we reiterate that the computation based on the ideal security model does not use a trusted party).

Most of the studies of SMC problems are based on the ideal security model. It is observed that achieving this kind of security is not difficult, but achieving it efficiently is. According to the theoretical SMC studies, all of the SMC problems can be solved in theory using the circuit evaluation protocol [18]. But using this general solution for special cases of multi-party computation can be impractical; special solutions should be developed for special cases for efficiency reasons. Motivated by this observation, researchers started to look for special solutions for each specific SMC problem. A number of researchers have proposed various solutions to the Private Information Retrieval (PIR) problem [9, 8, 21, 10, 22, 6, 16]; Du and Atallah have proposed the solutions to the privacy-preserving statistical analysis, scientific computation, and computational geometry problems [11, 15, 13, 2, 14]; Lindell and Pinkas have proposed the privacy-preserving data mining problem [23]. However, those solutions, although very elegant, are still not efficient enough for practical uses. Practical solutions need to be developed. However, whether practical solutions based on the ideal security model exist or not is still unknown.

Instead of following the traditional path to find practical solutions, we ask these questions: *Why do we have to achieve this kind of ideal security? Is ideal security really necessary in practice?* In the real world, ideal security is of course preferred, but if the ideal security is too expensive to achieve, people might prefer low-cost solutions that can achieve security at an "acceptable" level. In another words, sacrificing some security or disclosing some limited information about the private data for a better performance is often acceptable in practice. For example, if the private information is an array of numbers, disclosing some statistical information about these numbers might be acceptable as long as the actual raw data are not disclosed.

Therefore, we propose a new security paradigm: to study the secure multi-party computation problems based on an *acceptable* security model. The new paradigm is illustrated in Figure 1. We refer the new problem as the Practical Secure Multi-party Computation (PSMC) problem. PSMC problem deals with computing any function on any input, in a distributed network where each participant holds one of the inputs, ensuring that only a limited amount of information is revealed to a participant in the computation.

As the first step towards defining *acceptable* security, we use the following informal definition in this paper; a more formal definition is still under development.

> A protocol achieves acceptable security, if an adversary can only narrow down all the possible values of the secret data to a domain with the following properties:
>
> 1. The number of values in this domain is infinite, or the number of values in this domain is so large that a brute-force attack is computationally infeasible.
>
> 2. The range of the domain is acceptable for the application. The definition of the acceptable range depends on specific applications.

We let the definition of the acceptable range be application dependent because a range acceptable to one application might be unacceptable to others. For example, knowing that a number is within the domain of [0, 5] might not disclose important information for one application, but might disclose enough information for another application. Thus, this definition intrinsically implies that a solution satisfying this security definition should be adjustable in the sense the range of the domain disclosed to an adversary should be adjustable. Therefore users can adjust a protocol to satisfy their security needs with the minimum cost.

Based on this new definition, we studied a number of known SMC problems. Our results have shown that the performance of our new solutions improves significantly. To demonstrate how the new paradigm can lead to practical solutions, we describe the solutions to the basic computations that serves as the building blocks to many SMC problems. We will compare the new solutions with the existing results to understand the performance improvement achieved by our new solutions. In particular, We will demonstrate how the new paradigm can lead to practical solutions in two levels: the computation model level and the techniques levels. In the computation model level, we will describe the existing computation models that can serve our purpose. In the techniques level, we will demonstrate various techniques that have successfully achieved the acceptable security with a much better performance.

The paper is organized in the following way: Section 2 discusses the related work. Section 3 defines the scalar product problem that will be used throughout the paper. Then Section 4 and Section 5 demonstrate the approaches we have developed in achieving adjustable and acceptable security, with Section 4 focusing on the approach at the computation model level, and Section 5 focusing on more concrete techniques level. Finally Section 6 concludes the paper and lays out future work.

2. RELATED WORK

Secure Multi-party Computation. The history of the multi-party computation problem is extensive since it was introduced by Yao [25] and extended by Goldreich, Micali, and Wigderson [19], and by many others: Goldwasser [20] predicts that "the field of multi-party computations is today where public-key cryptography was ten years ago, namely an extremely powerful tool and rich theory whose real-life usage is at this time only beginning but will become in the future an integral part of our computing reality".

Goldreich states in [18] that the general secure multi-party computation problem is solvable in theory. However, he also points out that using the solutions derived by these general results for special cases of multi-party computation, can be impractical; special solutions should be developed for special cases for efficiency reasons.

Private Information Retrieval
Among various multi-party computation problems, the *Private Information Retrieval (PIR)* problem has been widely studied. PIR solutions provide a user with information from a database in a private manner, namely, the user's query information is not disclosed to the database. In this model,
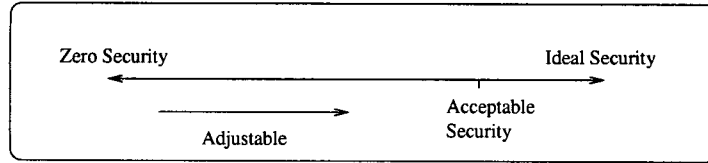
Figure 1: Acceptable Security

the database is viewed as an n-bit string x, and the user retrieves the i-th bit x_i out of x, while giving the database no information about the query i. The notion of PIR was introduced in [9], and was extensively studied in the literature [9, 8, 21, 10, 22, 6, 16].

PIR is not concerned with the privacy of the database. The problem of *Symmetrically Private Information Retrieval (SPIR)* is an extension of PIR where the database's privacy is also addressed. SPIR problem adds an extra requirement that the user, on the other hand, cannot obtain any information about the database in a single query except for the result. SPIR was first introduced in [17], could be solved using PIR protocols with a small complexity overhead [17, 24].

Privacy-Preserving Data Mining.
The privacy-preserving data mining problem is another specific secure multi-party computation problem that has been discussed in the literature. Recently, two different privacy-preserving data mining problems were proposed. In Lindell and Pinkas' paper [23], the problem is defined as this: Two parties, each having a private database, want to jointly conduct a data mining operation on the union of their two databases. How could these two parties accomplish this without disclosing their database to the other party, or any third party. A solution based on oblivious transfer protocol is presented in the paper. In Agrawal and Srikant's paper [1], the privacy-preserving data mining problem is defined as this: Alice is allowed to conduct data mining operation on a private database owned by Bob, how could Bob prevent Alice from accessing precise information in individual data records, while Alice is still able to conduct the data mining operations? This problem, although very interesting, does not fit into the secure multi-party computation framework. A data perturbation method is used in this work.

Selective Private Function Evaluation.
Selective Private Function Evaluation (SPFE) was introduced in [7]. In this problem, a client interacts with one or more servers holding copies of a database $x = x_1, \ldots, x_n$ in order to compute $f(x_{i_1}, \ldots, x_{i_m})$, for some function f and indices $i = i_1, \ldots, i_m$ chosen by the client. Ideally, the client must learn nothing more about the database than $f(x_{i_1}, \ldots, x_{i_m})$, and the servers should learn nothing. Various approaches for constructing sublinear-communication SPFE protocols were presented in [7], both for the general problem and for special cases of interest, such as the statistical functions.

Other Specific SMC problems.
Secure Multi-party Computation problems also exist in many other computation domains. New problems can emerge if we

combine the privacy requirements with a specific computation domain. Du and Atallah have studied a number of new SMC problems [11, 15], including the privacy-preserving scientific computation problem [13], the privacy-preserving statistical analysis problem [14], the privacy-preserving computational geometry problem [2], and the privacy-preserving database query problem [12]. Like the other SMC studies, all these works aim at achieving the *ideal* security.

3. SCALAR PRODUCT PROBLEM
To demonstrate our new paradigm, we will describe various techniques that support our new paradigm, as well as discuss how those techniques work. To make it easy to compare these techniques, we will use a single problem throughout this paper. However, as we will mention later, our techniques can solve a class of problems, not just this specific problem. The problem is called *Scalar Product Problem*, it is defined in the following:

PROBLEM 1. *(Scalar Product Problem) Alice has a vector $X = (x_1, \ldots, x_n)$ and Bob has a vector $Y = (y_1, \ldots, y_n)$. Alice (but not Bob) is to get the result of $u = X \cdot Y + v$ where v is a random scalar known to Bob only.*

The purpose of Bob's random v is as follows: If $X \cdot Y$ is a partial result that Alice is not supposed to know, then giving her $X \cdot Y + v$ prevents Alice from knowing the partial result (even though the scalar product has in fact been performed); later, at the end of a multiple-step protocol, the effect of v can be effectively "subtracted out" by Bob without revealing v to Alice.

4. COMPUTATION MODELS
In the study of the Secure Two-party problems, A Two-party model is usually used because it is an ideal model in terms of the security it provides. The Two-party model consists of just two parties (Figure 2.a), namely, Alice and Bob will conduct the computation totally by themselves without the help from any third party. If the two-party model can provide a practical solution, we do not need another model. However, according to our past experience, efficient solutions for this model are usually difficult to find.

Since the goal of our research is to achieve practical security, we will not limit ourselves to the Two-party model only; we would like to investigate other computation models, and study whether practical solutions can be achieved for those models. We understand that some computation models might achieve a weaker security than the Two-party model, but if the performance gain outweighs the security
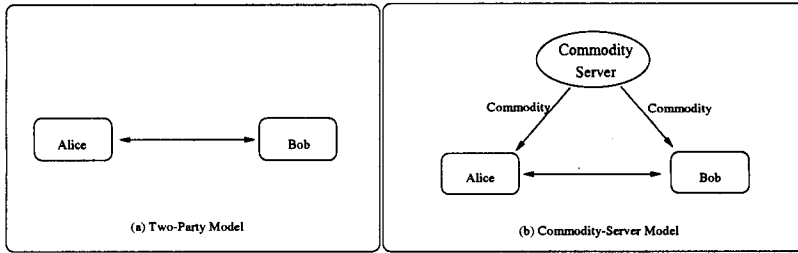
Figure 2: Computation Models

loss, if the security loss is acceptable in certain specific situations, the solutions based on those models are more likely to be adopted in practice.

One of the interesting models we have studied is the Commodity-Server Model depicted in Figure 2.b. The model introduces an extra server, the commodity server, belonging to a third party. The only requirement posed on the third server is that it cannot collude with either participants. This server has a few appealing features: First, this party does not participate in the computation between participants, but it does provide data for them to hide their private data. Second, the data provided by the commodity server does not depend on the participants' private data, so the commodity server does not need to know those private data. This feature gets the commodity server out of the picture of any future liability issues in case some participant's private data is disclosed somehow. Furthermore, this feature makes it easy to find such a server because not much trust is needed for a commodity server. With these features, the commodity server can generate independent data off-line, and sell them as commodities to the participants (whence the name "commodity server"). It should be pointed out that the commodity server should not collude with any parties, otherwise, the model simply becomes the Two-Party model. In reality, finding such a commodity server is very feasible.

The commodity server model was first proposed by Beaver [4, 5], and has been used for solving Private Information Retrieval problems in the literature [4, 5, 10]. In the following, we will describe a solution to the scalar product problem based on this model. It should be pointed out that the performance of this protocol is much more efficient than the other protocols that are based on the Two-party model. Some of the protocols will be described later in the paper.

PROTOCOL 1. *(Scalar Product Protocol–Commodity Server Approach)*

Inputs: *Alice has a vector $X = (x_1, \ldots, x_n)$, and Bob has a vector $Y = (y_1, \ldots, y_n)$.*

Outputs: *Alice (but not Bob) gets $X \cdot Y + v$, and Bob gets v.*

1. The Commodity Server generates a pair of random vectors R_a and R_b, and let $r_a + r_b = R_a \cdot R_b$, where r_a and r_b are randomly generated. Then the server sends R_a and r_a to Alice, sends R_b and r_b to Bob.

2. Alice sends $X' = X + R_a$ to Bob, and Bob sends $Y' = Y + R_b$ to Alice.

3. Bob generates a random number v', and computes $X' \cdot Y + v'$, then sends the result to Alice. Bob also lets $v = v' - r_b$.

4. Alice computes $(X' \cdot Y + v') - (R_a \cdot Y') + r_a = X \cdot Y + (v' - R_a \cdot R_b + r_a) = X \cdot Y + (v' - r_b) = X \cdot Y + v$.

THEOREM 1. *Assuming that all of the random numbers are generated from the real domain, Protocol 1 is secure such that Bob does not learn $X[i]$ for any i, and Alice does not learn $Y[i]$ for any i, where $X[i]$ and $Y[i]$ represent the vector's ith element, respectively.*

PROOF. **Claim 1**: Protocol 1 does not allow Bob to learn $X[i]$.

Since $X' = X + R_a$ is all what Bob gets, because of the randomness, and the secrecy of R_a, Bob cannot find out $X[i]$ for any i.

Claim 2: Suppose that protocol 1 does allow Alice to learn Y, namely Alice can find a vector X', such that by sending X' to Bob who follows the protocol, Alice can find out $Y[i]$ for some i. Formaly speaking, if the protocol is not secure, then there exists a deterministic algorithm such that, given X', for any Y, if the inputs of the algorithm are X', $Y' = Y + R_b$ and $Z = X'Y + v'$, the algorithm outputs $Y[i]$ for some i.

Next, we construct an arbitrary $\hat{Y}$ with $\hat{Y}[i] \neq Y'[i]$. Let $\hat{R}_b = Y' - \hat{Y}$ and $\hat{v}' = X'(Y - Y') + v'$. Because $\hat{Y} + \hat{R}_b$ still equals to Y', and $X'\hat{Y} + \hat{v}'$ still equals to Z, the algorithm should still output $Y[i]$. According to the deterministic algorithm, the output should be $\hat{Y}[i] = Y[i]$. Since $\hat{Y}[i] \neq Y[i]$, we have a contradiction, in other words, such deterministic algorithm does not exist.

From claim 1 and claim 2, we conclude that protocol 1 does not allow Bob to learn X, and it does not allow Alice to learn Y either. □

4.1 Security and Complexity Analysis.

If the random numbers are not generated from the real domain, Alice might get some information about Y. For example, if the elements of Y are in the domain of $[0, 100]$,

and we also know the random numbers are generated from $[0, 200]$ domain, then if an element of vector $Y + R_b$ is 250, we know the original element in vector Y is bigger than 50.

It should also be noted that our protocol does not deal with the situation where one party lies about its input. For example, instead sending $Y + R_b$, Bob sends $\hat{Y} + R_b$, where $\hat{Y}$ is an arbitrary vector. In that case, neither of them can get correct results, but as we have shown, neither of them can gain information about the other party's private input either.

In the third step, the purpose of the extra random v' is for Bob to protect the actual value of $X' \cdot Y$. If Alice is allowed to know the actual result of $X' \cdot Y$, Alice could learn partial information about Y or probably the whole information about Y if the same Y is used to compute several scalar products.

The above protocol is based on the assumption that the commodity server should not collude with either of Alice and Bob. However, we can improve security of the protocol. One way to improve it is that, instead of using only one commodity server, we can use multiple commodity severs. For example, if we use m number of commodity servers, we can cut the vectors X and Y to m pairs of smaller vectors, by using the above protocol, each commodity server can help to compute the scalar product of one of the pairs among m smaller vectors, the sum of the scalar products of these m pairs is the final scalar product of X and Y.

The communication cost of this approach is just $4n$. With this cost, the solution is efficient enough to become practical.

5. DATA DISGUISING TECHNIQUES
In this section, we discuss a number of specific techniques that are useful for data disguising. Some of the techniques do allow information disclosure to some extent; therefore, for each technique, we discuss what information is disclosed, and what kind of privacy is achieved. More importantly, we will discuss the properties of each technique, and the computation each technique can support. If the data disguise does not support any computation, then it is basically useless because our final goal is to conduct some computation. Therefore, most of the encryption schemes, except the homomorphic encryption scheme, are not of much use because they do not support computation on the encrypted data.

5.1 Linear Transformation Disguise
We define X_1 as a vector consisting of the first $n/2$ elements of the vector X; we define X_2 as a vector consisting of the second $n/2$ elements of X. Similarly we define Y_1 and Y_2 for vector Y. Note that $XY = X_1 Y_1 + X_2 Y_2$. To compute $u = X \cdot Y + v$, Alice can send X_1 to Bob while Bob sends Y_2 to Alice; Alice can then compute $u = X_1 Y_1$ and Bob can compute $v = -X_2 Y_2$. This does not require each side to disclose all data to the other side, but this scheme is unacceptable because each party has disclosed half of their private raw data.

To solve this problem, we can transform X (resp. Y) to another vector X' (resp. Y'), such that disclosing partial information about X' does not allow anybody to derive the

raw data of X. A linear transformation would achieve this goal, namely if we let $X' = XM$, where M is an invertible $n \times n$ matrix, disclosing half of the data of X' does not allow any one to derive the original raw data of X. Based on this observation, we derive our protocol (for the purpose of simplicity, we assume n is even; this can be achieved by padding the vectors with a 0 when n is odd):

PROTOCOL 2. *(Two-Party Scalar Product)*

1. *Alice and Bob jointly generate a random invertible $n \times n$ matrix M.*

2. *Alice lets $X' = XM$, and divides X' equally to two vectors X_1' and X_2'. Alice sends X_2' to Bob.*

3. *Bob lets $Y' = M^{-1}Y$, and divides Y' equally to two vectors Y_1' and Y_2'. Bob sends Y_1' to Alice.*

4. *Alice computes $u = X_1' Y_1'$.*

5. *Bob computes $v = -X_2' Y_2'$.*

It is easy to see that the above protocol achieves $u = X'Y' + v = XY + v$.

Security and Complexity Analysis.
To analyze how secure this protocol is, we need to find out how much Alice and Bob know about each other's information. According to this protocol, Bob knows $n/2$ data items in vector XM. Let us consider $X = (x_1, \ldots, x_n)$ as n unknown variable, and XM as a linear system of equations on these n unknown variables. If Bob knows all of the n equations, Bob can easily solve this linear system, and recover the values in X. However, in this protocol, Bob only knows $n/2$ of the equations. Theoretically, if x_i's are in real number domain, based on these $n/2$ equations, there are infinite number of solutions to the n unknown variables. Therefore, although Bob learns $n/2$ linear combination of the data, it is impossible for Bob to learn actual values of the data in vector X.

The above protocol is secure if the values of the data items in vector X and Y are real numbers. However, in situations where these values are integers or just 0 or 1, sometimes $n/2$ equations might be enough for Bob to find the actual values of x_i's if n is not very big. Therefore, the above protocol is only secure in the real number domain.

We should be careful if we want to reuse X or Y for another scalar product; otherwise the security might be compromised. For example, in order to compute another scalar product XZ, Alice needs to disclose another $n/2$ equation to Bob. If these $n/2$ equations are linear independent to the first $n/2$ equations disclosed to Bob during the computation of XY, Bob now has n equations, and therefore can solve these n equations to find out the actual values of X. To avoid this type of security compromise, we should use the same matrix M when X (or Y) is reused.

The communication cost of this protocol is n, which is as good as the scalar product computation in the non-secure

situation, namely one party sends its vector in plain text to the other party. The computation cost is $O(n^3)$ due to the computation of M^{-1}. However, there is a way to improve the computation cost: instead of using a matrix of size $n \times n$, we can use n/m matrices, each of which has size $m \times m$, namely we cut the vector X and Y to n/m pairs of small vectors, the sum of the scalar products of these n/m pairs is the final scalar product of X and Y. For the scalar product of each pair, we can use the above protocol with a random matrix of size $m \times m$. Therefore, the computation cost is $O(n/m * (m)^3) = O(nm^2)$. If we choose a smaller m, the computation cost is improved, but more information will be disclosed. Thus, the security level can be adjusted by the users who will decide the tradeoff between performance and security.

5.2 $(Z + V)$-Disguise

If a secret data item Z needs to be stored at Alice's place without being disclosed to Alice, a simple solution is to give Alice $Z + V$, where V is a data item consisting of random numbers, and Alice does not know V. In a finite domain, $Z + V$ can perfectly hide Z; however, in an infinite domain, $Z + V$ can reveal some information about Z, although it is still impossible to recover the value of Z. The bigger the V is, the more secure the disguise will be. Therefore by adjusting the size of the V, we can adjust the degree of the disguise.

Regardless of how a data item is disguised, the data should be able to be used in the computation we want to achieve; otherwise, if the disguise is the sole purpose, we will just use the encryption schemes, which are the best ways to disguise data. In the next few subsections, we will discuss how the basic computations can be conducted based on this $Z + V$ data disguise scheme.

We will start from the situation where Alice has a data item X, and Bob has a data item Y; they want to achieve $f(X, Y)$, where f represents a basic computation. Since some times knowing the result of $f(X, Y)$ and one of the inputs allows one to derive the value of the other inputs, we will not disclose the result of $f(X, Y)$ to anyone. Therefore, in addition to using the $(Z + V)$-Disguise technique to conduct the computation, we will use the same technique to hide the result as well, namely, we will let Alice learn $f(X, Y) + V$ and let Bob learn V.

The basic computations in our discussion include the *addition*, *multiplication*, and *inverse*. Most of the more complicated computations can be based on these basic computations.

PROBLEM 2. *Two secrets, Z_1 and Z_2, are shared by Alice and Bob in such a way that Alice only knows $Z_1 + V_1$ and $Z_2 + V_2$, and Bob knows V_1 and V_2; however neither knows either Z_1 or Z_2. Alice and Bob want to conduct the following computations (In this problem Z_i and V_i could both be, if applicable, numbers, arrays of numbers, or Matrices):*

- $Z_1 + Z_2$: *Alice gets $(Z_1 + Z_2) + V$, and Bob gets V.*

- $Z_1 \cdot Z_2$: *Alice gets $Z_1 \cdot Z_2 + V$, and Bob gets V.*

- $U \cdot Z_1$: *Alice gets $U \cdot Z_1 + V$, and Bob gets U and V. Because Alice does not know U, the computation can still disguise the secret Z_1 even if V is zero. In fact, when we use the computation, sometimes, we will let V be zero, and let V be non-zero at other times.*

- Z_1^{-1}: *Alice gets $Z_1^{-1} + V$, and Bob gets V.*

- $(Z_1 + Z_2)^{-1}$: *Alice gets $(Z_1 + Z_2)^{-1} + V$, and Bob gets V. This computation can be obtained from the solutions to the Z_1^{-1} computation because Alice knows $(Z_1 + Z_2) + (V_1 + V_2)$, and Bob knows $(V_1 + V_2)$, thus we can achieve $(Z_1 + Z_2)^{-1}$ computation using the same method.*

- $\log Z_1$: *Alice gets $\log Z_1 + V$, and Bob gets V.*

- $\log(Z_1 + Z_2)$: *Alice gets $\log(Z_1 + Z_2) + V$, and Bob gets V. Similar to the computation of $(Z_1 + Z_2)^{-1}$, the computation of $\log(Z_1 + Z_2)$ is the same as the computation of $\log Z_1$.*

Next, we will describe the outline of each protocol. The most important protocol is the $(U \cdot Z)$-protocol, because most of the other protocols can be derived from this protocol. Therefore, we will only describe this protocol in details, and give a brief outline of the other protocols.

PROTOCOL 3. $(Z_1 + Z_2)$-protocol: *This protocol is trivial. Alice just needs to compute $(Z_1 + V_1) + (Z_2 + V_2)$, and gets $(Z_1 + Z_2) + (V_1 + V_2)$; Bob needs to compute $V = V_1 + V_2$. Therefore Alice has $(Z_1 + Z_2) + V$ and Bob has V.*

PROTOCOL 4. $(U \cdot Z)$-protocol: *Because this protocol is used as a building block by many other protocols described later, its performance is very important. We have developed several solutions for this protocol. Here, we first describe one solution based on the oblivious transfer protocol. Later in this section, we will describe another solution that is more efficient but less secure.*

1. *Alice and Bob agree on two numbers p and m, such that p^m is large enough.*

2. *Alice randomly generates $R_1, \ldots, R_m$, such that $\sum_{j=1}^{m} R_j = Z_1 + V_1$.*

3. *Bob randomly generates $r_1, \ldots, r_m$ such that $\sum_{j=1}^{m} r_j = V - UV_1$.*

4. *For each $j = 1, \ldots, m$, Alice and Bob conduct the following sub-steps:*

   (a) *Alice generates a secret random number k, $1 \leq k \leq p$.*

   (b) *Alice sends $(H_1, \ldots, H_p)$ to Bob, where $H_k = R_j$, and the rest of H_i's are random. Because k is a secret number known only to Alice, Bob does not know the position of R_j.*

   (c) *Bob computes $T_{j,i} = U \cdot H_i + r_j$ for $i = 1, \ldots, p$.*

(d) Using the 1-out-of-p Oblivious Transfer protocol, Alice gets $T_j = T_{j,k} = U \cdot R_j + r_j$, while Bob learns nothing about k.

5. Alice computes $u = \sum_{j=1}^{m} T_j = U \cdot (Z_1 + V_1) + (V - UV_1) = U \cdot Z_1 + V$.

Security and Complexity Analysis.

In the above protocol, Alice divides $Z_1 + V_1$ into m random pieces $R_1, \ldots, R_m$, and then gets back $U \cdot R_j + r_j$ for $j = 1, \ldots, n$. Because of the randomness of R_i and its position among other bogus data, Bob could not find out which one is R_i. Certainly, there is 1 out of p possibility that Bob can guess the correct R_i, but since $Z_1 + V_1$ is the sum of m such random pieces, the chance that Bob guesses the correct $Z_1 + V_1$ is 1 out p^m, which could be very small if we chose p^m to be large enough.

The communication cost of this protocol is $O(mpn)$. If we let $p = 2$ and $m = 128$ (so the brute-force attack has to conduct 2^{128} steps), the cost would be about $256n$. This solution is not a practical solution, although it is very secure except for the information disclosed because of the $(Z+V)$-Disguise. In the last section, we have described a solution based on the Commodity-Server model; the cost of that solution is only $4n$, very close to the optimum cost of n. In the next section, we are going to describe another solution based on the Two-Party Model. That new solution only has the communication cost of $2n$, but the computation cost on the other hand increases significantly.

PROTOCOL 5. $(Z_1 \cdot Z_2)$-protocol: This protocol is essentially an application of the $(U \cdot Z)$-protocol.

1. Bob randomly generates V, V_3, and V_4, such that $V = -V_1V_2 - (V_3 + V_4)$.

2. Alice computes $W_1 = (Z_1 + V_1) \cdot (Z_2 + V_2) = Z_1Z_2 + Z_1V_2 + Z_2V_1 + V_1V_2$.

3. Using the $(U \cdot Z)$-protocol on $Z_1 + V_1$ and V_2, Alice can get $W_2 = V_2Z_1 + V_1V_2 + V_3$.

4. Using the $(U \cdot Z)$-protocol on $Z_2 + V_2$ and V_1, Alice can get $W_3 = V_1Z_2 + V_1V_2 + V_4$.

5. Alice computes $W_1 - W_2 - W_3 = Z_1Z_2 + (-V_1V_2 - (V_3 + V_4)) = Z_1Z_2 + V$

PROTOCOL 6. (Z^{-1})-protocol: This protocol assumes that Z_1 is a non-zero number or a non-singular matrix. By using the $(U \cdot Z)$-protocol, Alice can get $U \cdot Z_1$, and Bob gets U. Alice can then compute $(U \cdot Z_1)^{-1}$. They use the $(U \cdot Z)$-protocol again, and this time Alice gets $(U \cdot Z_1)^{-1} \cdot U) + V = Z_1^{-1} + V$ and Bob gets V.

PROTOCOL 7. $\log Z_1$-protocol: This protocol assumes that Z_1 is a positive number. First, Bob generates a random number U and let $V = \log U$. Then by using the $(U \cdot Z)$-protocol, Alice can get $U \cdot Z_1$. Then Alice computes $\log(U \cdot Z_1) = \log Z_1 + \log U = \log Z_1 + V$.

5.3 Polynomial Function Disguise

Another way to hide a secret $Z = (z_1, \ldots, z_n)$ is to use the polynomial function of degree k. Each z_i is hidden in $f(x) = a_kx^k + \ldots + a_1x + a_0$, with the parameters $a_0, \ldots, a_k$ being unknown to the person who holds $f(z_i)$. To find out the value of z_i, one needs to find out the $k + 1$ parameters.

A good property of this kind of disguise is that it can preserve the order among $z_1, \ldots, z_n$, if we know the domain of the z_i's. Therefore by comparing $f(z_i)$ for $i = 1, \ldots, n$, we can find the minimum, maximum, and the total order among these n secret numbers. We will describe a *FindMin* protocol to illustrate the use of the polynomial function disguise. Comparing to the solution proposed in [3], this protocol is significantly more efficient.

PROBLEM 3. *(FindMin Problem) Alice has an array of numbers $X = (z_1 + v_1, \ldots, z_n + v_n)$, and Bob has an array of numbers $V = (v_1, \ldots, v_n)$. They want to find out the m, such that $z_m = min\{z_i | i = 1, \ldots, n\}$. Nobody should know the value of z_i's.*

To solve this problem, Alice and Bob divide their data to n/m groups, with each group containing m numbers. The idea is to first find the minimum number within each group, and thus get n/m numbers. Then we recursively divide these n/m numbers to groups, and find the minimum number within each group, until there is only one number left. This number is the smallest among all these n numbers.

So how to find the minimum number among m numbers? Without the loss of generality, we suppose these m numbers to be $z_1, \ldots, z_m$. We will let Bob generate a k-degree function $f(x)$, and Alice can use our scalar product protocol to get the value of $f(z_i)$ (see footnote[1]), for $i = 1, \ldots, m$. $f(x)$ is generated in a way such that $f(z_i) \leq f(z_j)$ if and only if $z_i \leq z_j$. This can be achieved if we know the possible domain of those z_i's. For instance, if we know that all z_i's are positive, we can find such a $f(x)$. Because of this order-preserving property, Alice can find the smallest z_i by herself without knowing the value of z_i's.

Security and Complexity Analysis.
The communication cost and the computation cost of the above solution are decided by the number of the invocation of the scalar product protocol, and the degree k of the function $f(x)$. The number of the invocation equals to the number of the recursive iterations, which equals to $\log_m n$. By increasing the value of m, we can reduce the actual cost of the solution; however, when the value of m becomes bigger, the security becomes weaker if k keeps to be the same. This is because when a k-degree polynomial function is used to disguise m numbers, the security is decided by the difference of k and m. Intuitively speaking, in our solution, we are using k random numbers to hide m numbers. Therefore, the bigger the value of k, the more secure our solution is. Let us think about an extreme case when $k = 1$. It means

---
[1] Since $z_i = (z_i + v_i) - v_i$, if we expand the function $f(z_i)$, we will find out that the evaluation of $f(z_i)$ is actually a scalar product, with Alice knowing $z_i + v_i$ and Bob knowing v_i.

all these m numbers are actually hidden by a single number. Once this number is known, all these numbers will be compromised. On the other hand, the bigger the value of k, the more expensive the computation and computation cost will be. Therefore, we can adjust the security and the cost of our solution by adjusting the value of k.

Comparing to the *FindMin* protocol we developed before [3], this protocol is much more efficient, but it is less secure because several information about z_i's are disclosed. For example, Since Alice is conducting the comparison, apart from knowing the final result, the minimum elements, she also knows which z_i is bigger.

5.4 Other Disguise Techniques

There are many other disguise methods, and their applications are problem-dependent. We have successfully used them to solve a number of specific secure two-party computation problems. We briefly summarize these methods in the following:

- Permutation: If the data is a list of numbers, sometimes, permute the order of these numbers might be a useful technique to disguise the data. The technique is used in [14] to hide the contents of a vector.

- Adding bogus data: If the data is a database, sometimes, adding significant amount of bogus records into the database can make it not much useful if one does not know which records are bogus records. Therefore, by adding bogus data into a database, we can achieve the data disguise to some degree. We are currently exploiting this technique to achieve data disguise purpose in another research project.

6. CONCLUSION AND FUTURE WORK

We propose a new security paradigm for the secure multi-party computation studies. Instead of achieving the ideal security like most of the studies of SMC problems, we propose a different goal, to achieve acceptable (and adjustable) security. We have shown that various data disguise techniques and the commodity-server computation model can help improve the performance of the solutions with certain degree of security sacrifice. We have also shown that our techniques are adjustable towards either more efficient or more secure, namely users can decide the degree of tradeoff between the performance and the security. If the security is very important to the users, and no information should be disclosed, they can adjust the corresponding parameters in our solutions to increase the security level. If the users decide that disclosing partial information is still acceptable, they can choose a parameter that can efficiently achieve their requirements. We believe this new model, which takes into the consideration the requirements from the practice, can lead to the promising application of the secure multi-party computation problems.

The work based on this new security paradigm is far from complete, we need to develop more efficient data disguise .techniques and more computation models that can be used to solve various specific ·secure multi-party (or two-party) problems. In addition, in our future work, we also need to find a way to quantify the security achieved in each protocol, so we can compare protocols regarding to the level of security they can achieve.

7. ACKNOWLEDGES

8. REFERENCES

[1] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proceedings of the 2000 ACM SIGMOD on Management of Data*, pages 439–450, Dallas, TX USA, May 15 - 18 2000.

[2] M. J. Atallah and W. Du. Secure multi-party computational geometry. In *WADS2001: 7th International Workshop on Algorithms and Data Structures*, pages 165–179, Providence, Rhode Island, USA, August 8-10 2001.

[3] M. J. Atallah, F. Kerschbaum, and W. Du. Private file comparision. Technical report, Purdue University, 2001.

[4] D. Beaver. Commodity-based cryptography (extended abstract). In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, El Paso, TX USA, May 4-6 1997.

[5] D. Beaver. Server-assisted cryptography. In *Proceedings of the 1998 New Security Paradigms Workshop*, Charlottesville, VA USA, September 22-26 1998.

[6] C. Cachin, S. Micali, and M. Stadler. Computationally private information retrieval with polylogarithmic communication. *Advances in Cryptology: EUROCRYPT '99, Lecture Notes in Computer Science*, 1592:402–414, 1999.

[7] R. Canetti, Y. Ishai, R. Kumar, M. K. Reiter, R. Rubinfeld, and R. N. Wright. Selective private function evaluation with applications to private statistics (extended abstract). In *Proceedings of Twentieth ACM Symposium on Principles of Distributed Computing (PODC)*, 2001.

[8] B. Chor and N. Gilboa. Computationally private information retrieval (extended abstract). In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, El Paso, TX USA, May 4-6 1997.

[9] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *Proceedings of IEEE Symposium on Foundations of Computer Science*, Milwaukee, WI USA, October 23-25 1995.

[10] G. Di-Crescenzo, Y. Ishai, and R. Ostrovsky. Universal service-providers for database private information retrieval. In *Proceedings of the 17th Annual ACM Symposium on Principles of Distributed Computing*, September 21 1998.

[11] W. Du. *A Study of Several Specific .Secure Two-party Computation Problems*. PhD thesis, Purdue University, West Lafayette, Indiana, 2001.

[12] W. Du and M. J. Atallah. Protocols for secure remote database access with approximate matching. In *7th ACM Conference on Computer and Communications Security (ACMCCS 2000), The First Workshop on Security and Privacy in E-Commerce*, Athens, Greece, November 1-4 2000.

[13] W. Du and M. J. Atallah. Privacy-preserving cooperative scientific computations. In *14th IEEE Computer Security Foundations Workshop*, pages 273–282, Nova Scotia, Canada, June 11-13 2001.

[14] W. Du and M. J. Atallah. Privacy-preserving statistical analysis. In *Proceedings of the 17th Annual Computer Security Applications Conference*, pages 102–110, New Orleans, Louisiana, USA, December 10-14 2001.

[15] W. Du and M. J. Atallah. Secure multi-party computation problems and their applications: A review and open problems. In *New Security Paradigms Workshop*, pages 11–20, Cloudcroft, New Mexico, USA, September 11-13 2001.

[16] Y. Gertner, S. Goldwasser, and T. Malkin. A random server model for private information retrieval. In *2nd International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM '98)*, 1998.

[17] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting data privacy in private information retrieval schemes. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, Dallas, TX USA, May 24-26 1998.

[18] O. Goldreich. Secure multi-party computation (working draft). Available from http://www.wisdom.weizmann.ac.il/home/oded/public_html/foc.html, 1998.

[19] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, pages 218–229, 1987.

[20] S. Goldwasser. Multi-party computations: Past and present. In *Proceedings of the 16th Annual ACM Symposium on Principles of Distributed Computing*, Santa Barbara, CA USA, August 21-24 1997.

[21] Y. Ishai and E. Kushilevitz. Improved upper bounds on information-theoretic private information retrieval (extended abstract). In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, Atlanta, GA USA, May 1-4 1999.

[22] E. Kushilevitz and R. Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *Proceedings of the 38th Annual IEEE Computer Society Conference on Foundation of Computer Science*, Miami Beach, Florida USA, October 20-22 1997.

[23] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Advances in Cryptology - Crypto2000, Lecture Notes in Computer Science*, volume 1880, 2000.

[24] M. Naor and B. Pinkas. Oblivious transfer and polynomial evaluation (extended abstract). In *Proceedings of the 31th ACM Symposium on Theory of Computing*, pages 245–254, Atanta, GA, USA, May 1-4 1999.

[25] A. C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, 1982.