Privacy-Preserving Cooperative Scientific Computations *

Wenliang Du and Mikhail J. Atallah
Department of Computer Sciences and
Center for Education and Research in Information Assurance and Security
Purdue University, 1315 Recitation Building
West Lafayette, IN 47907
Email: {duw,mja}@cs.purdue.edu

Abstract

The growth of the Internet has triggered tremendous opportunities for cooperative computation, in which multiple parties need to jointly conduct computation tasks based on the private inputs they each supply. These computations could occur between mutually untrusted parties, 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. Today, to conduct such a computation, one must usually know the inputs from all the participants; however if nobody can be trusted enough to know all the inputs, privacy will become a primary concern.

Linear systems of equations problem and linear least-square problem problems are two important scientific computations that involve linear equations. Solutions to these problems are widely used in many areas such as banking, manufacturing, and telecommunications. However, the existing solutions do not extend to the privacy-preserving cooperative computation situation, in which the linear equations are shared by multiple parties, who do not want to disclose their data to the other parties.

In this paper, we formally define these specific privacy-preserving cooperative computation problems, and present protocols to solve them.

1 Introduction

The growth of the Internet has triggered tremendous opportunities for cooperative computation, in which multiple parties need to jointly conduct computation tasks based on the private inputs they each supply. These computations could occur between mutually untrusted parties, 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. Today, to conduct such a computation, one must usually know inputs from all the participants; however if nobody can be trusted enough to know all the inputs, privacy will become a primary concern. For example, consider the following applications:

Two financial organizations plan to cooperatively work on a project for mutual benefit. Each of the organizations would like its own requirements being satisfied (usually, these requirements are modeled as linear equations). However, the requirements includes their projects of the likely future evolution of certain commodity prices, interest and inflation rates, economic statistics, and customers' portfolio holdings. These are valuable proprietary data that nobody is willing to disclose to other parties, or even to a "trusted" third party. How could these two financial organizations cooperate on this project?

Two companies A and B are investigating an opportunity for a partnership. Company A's goal is to optimize the cost of a manufacturing process. As part of the partnership, company B will conduct part of the process. Because of this, A does not know B's constraints on that part of the process, unless B tells A, nor does B know A's constraints. Usually, the constraints reflect the information about the company's resource, strategic plans, cost information, and business decisions. They are so critical that both companies try every measure to protect them. Considering that the partnership is not formed yet, B is afraid that, if the partnership eventually falls through, the information it provides to A might be used by A for B's disadvantage. With such a concern, B really does not feel comfortable to give its information to any other company, neither does A. How could these two companies find out the benefit of a potential partnership without risking their private information?

The above examples, without the privacy concerns,

*Portions of this work were supported by Grant EIA-9903545 from the National Science Foundation, and by sponsors of the Center for Education and Research in Information Assurance and Security.

could usually be modeled as linear systems of equations problems or linear least squares problems [15]. These scientific computation problems have proved valuable for modeling many and diverse types of problems in planning, routing, scheduling, assignment, and design. Industries that make use of these problems and their extensions include banking, transportation, energy, telecommunications, and manufacturing of many kinds. Although these problems have been well studied in the literature, their current solutions rarely extend to the situation in which multiple parties want to jointly conduct the computations based on the private inputs. For instance, Alice has k linear equations in n unknown variables x_i; Bob has $n-k$ linear equations in the same n unknown x_i. Alice and Bob want to find the solution $(x_1, \ldots, x_n)$ that satisfies the combined n linear equations. We know how to solve the problem if Alice can give her equations to Bob or vice versa, because it is just a normal linear system of equations problem. However, if the equations owned by each party are so valuable proprietary data that neither party is willing to disclose to the other, the problem can no longer be solved using the traditional methods, such as Gaussian elimination and LU factorization, because these methods assume that one who conducts the computation knows all the inputs, an assumption that is not true any more in the privacy-preserving cooperative computation situation. We need to find solutions that allow Alice and Bob to jointly solve their combined n linear equations while not disclosing each person's private equations to the other.

Currently, to solve the above problems, a commonly adopted strategy is to assume the trustworthiness of the participants, or to assume the existence of a trusted third party. Such assumptions are quite strong and maybe infeasible, and clearly it is desirable to have solutions that do not rely on the complete trustworthiness of participants or third parties. Moreover, in certain situation, even though we could trust that the other parties will not use our private information against our wish, we cannot guarantee that their systems being secure enough to prevent our information from being stolen. On the other hand, from the trusted parties' point of view, in order to conduct such a cooperative computation, they have to carry the extra burden of securing other party's data. If a disgruntled employee or a security breach causes the compromise of the data, these trusted parties might face expensive lawsuits. Therefore, it is to the favor of every participants that nobody knows the other parties' secret information. Protocols that can support this type of joint scientific computations while protecting the participants' privacy are of growing importance.

In this paper, we introduce the privacy-preserving cooperative scientific computations (PPCSC) problem. The general definition of the PPCSC problem is that two or more parties want to conduct a scientific computation based on their private inputs, but neither party is willing to disclose its own input to anybody else (including a so-called trusted third party). We have further defined several specific PPCSC problems, including privacy-preserving cooperative linear system of equations (PPC-LSE) problem, and privacy-preserving cooperative linear least-square (PPC-LLE) problem, all of which involve a matrix.

There are several ways to share a matrix. Depending on how such a matrix is shared by Alice and Bob, or in another word how Alice and Bob cooperate with each other, the problems could appear in a variety of forms. Figure 1 describes three different types of cooperation.

Figure 1(b) depicts the homogeneous cooperation, in which each party provides its own equations; Figure 1(c) depicts the heterogeneous cooperation, in which both parties have to jointly specify each single equation; Figure 1(d) depicts the hybrid cooperation, in which both parties cooperate in an arbitrary way. (b) and (c) are more meaningful cooperations than (d) in real life, and they are two special cases of problem (d). We have developed a protocol to solve the problem (d): $(M_1 + M_2)x = b_1 + b_2$, where matrix M_1 and vector b_1 belong to one party, matrix M_2 and vector b_2 belong to the other party. At the end of the protocol, both parties know the solution x while nobody knows the other party's private inputs. Based on this protocol and the similar techniques, we have solved PPC-LSE problems and PPC-LLE problems.

The generalization of the PPCSC problem is referred to as Secure Multi-party Computation problem (SMC) in the literature [22]. Generally speaking, a secure multi-party computation problem deals with computing any probabilistic 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 [8].

Goldreich states in [6] that the general secure multi-party computation problem is solvable in theory, but 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. Motivated by this assertion, we are interested in seeking special solutions to the specific PPCSC problem, solutions that are more efficient than the general theoretic solutions.

In the rest of this paper, the next subsection presents the related work. Section 2 presents formal definition of the privacy. Section 3 describes the PPC-LSE, PPC-LLE protocols and their applications. Section 4 discusses the efficiency of these protocols. Section 5 summarizes the paper and lays out some future work.

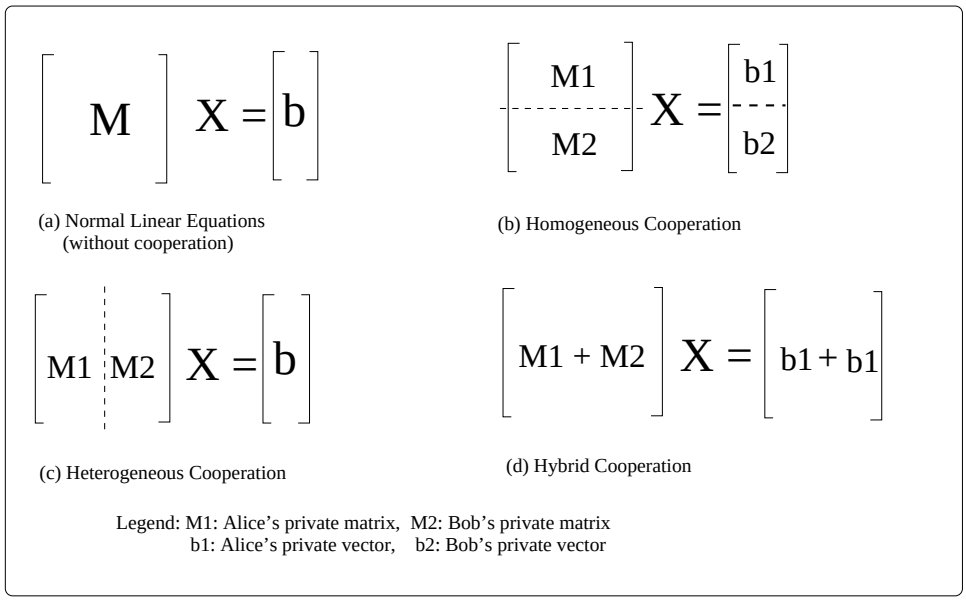

Figure 1. Various ways of cooperation

1.1 Related Work

The history of the multi-party computation problem is extensive since it was introduced by Yao [22] and extended by Goldreich, Micali, and Wigderson [18], and by many others. In the past, secure multi-party computation research has mostly been focusing on the theoretical studies, very few applied problems have been studied. Those few applied problems include Private Information Retrieval problem (PIR) [12, 3, 11, 10, 13, 17, 14, 9], Joint digital signature [21, 5] and joint decryption, elections over the Internet, electronic bidding [2], and privacy-preserving data mining [16, 1].

1-**out-of-**N **Oblivious Transfer**

An 1-out-of-N Oblivious Transfer protocol [7, 4] refers to a protocol where at the beginning of the protocol one party, Bob has N inputs $X_1, \ldots, X_N$ and at the end of the protocol the other party, Alice, learns one of the inputs X_I for some $1 \leq I \leq N$ of her choice, without learning anything about the other inputs and without allowing Bob to learn anything about I. An efficient 1-out-of-N Oblivious Transfer protocol was proposed in [19] by Naor and Pinkas. Their solution can achieve $O(m)$ communication complexity, where m is the security parameter (i.e. the length of a number that is hard to factor). This protocol serves as an important building block for our protocols, and the ideas of using the 1-out-of-N Oblivious Transfer protocol as building block are pioneered by Naor and Pinkas in [19].

2 Security Definition

The model for this work is that of general multi-party computation, more specifically between two *semi-honest* parties. Our formal definitions are according to Goldreich in [6]. We now present in brief the definition for general two-party computation of a functionality with *semi-honest* parties only. They are taken from [6].

Definition 2.1. *(privacy w.r.t. semi-honest behavior):* Let $f : \{0,1\}^* \times \{0,1\}^* \mapsto \{0,1\}^* \times \{0,1\}^*$ be a functionality, where $f_1(x,y)$(resp., $f_2(x,y)$ denotes the first (resp., second) element of $f(x,y)$, and Π be a two-party protocol for computing f. The `view` of the first (resp., second) party during an execution of Π on (x,y), denoted $VIEW_1^\Pi(x,y)$ (resp., $VIEW_2^\Pi(x,y)$), is $(x, r^1, m_1^1, \ldots, m_t^1)$ (resp., $(y, r^2, m_1^2, \ldots, m_t^2)$), where r^1 (resp., r^2) represents the outcome of the first (resp., second) party's internal coin tosses, and m_i^1 (resp., m_i^2) represents the i^{th} message it has received. The `output` of the first (resp., second) party during an execution of Π on (x,y), denoted $OUTPUT_1^\Pi(x,y)$ (resp., $OUTPUT_2^\Pi(x,y)$), is implicit in the party's view of the execution.

- We say that π `privately computes` f if there exist polynomial time algorithms, denoted S_1 and S_2 such that

  $\{(S_1(x, f_1(x,y)), f_2(x,y))\}_{x,y\in\{0,1\}^*}$
  $\equiv \{(VIEW_1^\Pi(x,y), OUTPUT_2^\Pi(x,y))\}_{x,y\in\{0,1\}^*}$

  $\{(f_1(x,y), S_2(y, f_2(x,y)))\}_{x,y\in\{0,1\}^*}$
  $\equiv \{(OUTPUT_1^\Pi(x,y), VIEW_2^\Pi(x,y))\}_{x,y\in\{0,1\}^*}$

where $\equiv$ denotes *computational indistinguishability.*

$VIEW_1^{\Pi}(x,y)$ and $VIEW_2^{\Pi}(x,y)$, $OUTPUT_1^{\Pi}(x,y)$ and $OUTPUT_2^{\Pi}(x,y)$ are related random variables, defined as a function of the same random execution.

3 Some Privacy-Preserving Cooperative Scientific Computations

In this section, we describe two related protocols for privacy-preserving cooperative scientific computation, including the protocols for the privacy-preserving cooperative linear system of equations (PPC-LSE) and privacy-preserving cooperative linear least-square problem (PPC-LLS). We assume a finite field F, and all computations are over this finite field, meaning that entries of matrices (or vectors) are elements of a finite field and addition and multiplication are defined with respect to that field. As a result, this assumption makes the scope of the computations somewhat different than the original computations. Such an assumption is made to achieve the privacy requirements according to Goldreich's definitions [6]. We believe that dropping this finite field assumption is possible if different privacy requirements (defined in an infinite domain) can be used.

3.1 Two Models of Cooperation

A common property of the above PPC-LSE and PPC-LLE problems is the combining knowledge of a matrix M and of a vector b. We have described in Figure 1 three different ways of combining knowledge, with (b) and (c) being the special cases of (d). However, in real life, cases (b) and (c) are more meaningful than (d) because they tend to model the ways of actual cooperations.

In the PPC-LSE and PPC-LLE problems, M and b usually represent a set of linear constraints. Sometimes the cooperating parties each has its own set of constraints, but sometimes they have to jointly specify each single constraint. Therefore we classify the cooperation to two basic models, the *heterogeneous* model and the *homogeneous* model.

Model 1. *(Homogeneous Model)* Alice has a matrix M_1 and a vector b_1; Bob has a matrix M_2 and a vector b_2. The size of M_1 is $m_1 \times n$, the size of M_2 is $m_2 \times n$; the lengths of the vectors b_1 and b_2 are m_1 and m_2, respectively. Alice and Bob want to solve

$$\begin{pmatrix} M_1 \\ M_2 \end{pmatrix} x = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

The model could be transformed to the the following form:

$$\left(\begin{pmatrix} M_1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ M_2 \end{pmatrix} \right) x = \begin{pmatrix} b_1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ b_2 \end{pmatrix}$$

Model 2. *(Heterogeneous Model)* Alice has a matrix M_1; Bob has a matrix M_2. The size of M_1 is $m \times n_1$, the size of M_2 is $m \times n_2$, where $n_1 + n_2 = n$. Alice and Bob both know a vector b of length m. They want to solve

$$\begin{pmatrix} M_1 & M_2 \end{pmatrix} x = b$$

The above linear equations could be transformed to the the following form:

$$\left(\begin{pmatrix} M_1 & 0 \end{pmatrix} + \begin{pmatrix} 0 & M_2 \end{pmatrix} \right) x = b + 0$$

Because both models are the special cases of the hybrid model (Figure 1 d), our solutions are developed for the hybrid model.

3.2 Linear System of Equations Problem

Problem 1. *(PPC-LSE)* Alice has a matrix M_1 and a vector b_1, and Bob has a matrix M_2 and a vector b_2, where M_1 and M_2 are $n \times n$ matrices, and b_1 and b_2 are n-dimensional vectors. Without disclosing their private inputs to the other party, Alice and Bob want to solve the linear equation

$$(M_1 + M_2)x = b_1 + b_2$$

The Protocol Without concerning about the privacy, a straightforward solution would be to ask one party (say Bob) to send his M_2 and b_2 to the other party, Alice. This however does not work if Bob is concerned about the privacy of his data. Bob cannot simply send M_1 and b_1 to Alice; he has to disguise the data in a way such that Alice cannot derive the original data from the disguised data.

Our solution is based on the fact that the solution to the linear equations $(M_1 + M_2)x = b_1 + b_2$ is equivalent to the solution to the linear equations $P(M_1 + M_2)QQ^{-1}x = P(b_1 + b_2)$. If Alice knows $M' = P(M_1 + M_2)Q$ and $b' = P(b_1 + b_2)$, she can solve the linear equation problem: $M'\hat{x} = b'$, and thus getting the final solution x, where $x = Q\hat{x}$. But how can Alice know M' and b' without being able to derive the value of M_2 and b_2? To solve this problem, Bob generates two invertible random $n \times n$ matrices P and Q. Then Alice and Bob use secure protocols (will describe them later) to get Alice (and only Alice) to learn the value of $P(M_1 + M_2)Q$ and $P(b_1 + b_2)$. However, Alice will not learn the value of PM_1Q, PM_2Q, Pb_1, Pb_2, much less P, Q, M_2, or b_2.

After Alice gets $M' = P(M_1 + M_2)Q$ and $b' = P(b_1 + b_2)$, she can solve the linear equations $M'\hat{x} = b'$ by herself, and then send the solution $\hat{x}$ to Bob, who can compute the

final solution $x = Q\hat{x}$. Finally Bob sends the solution to Alice. Although we do not prevent disruption of the entire computation if Alice or Bob misbehaves, we do allow Alice to detect the case where Bob learns the correct answer but does not allow Alice to learn the correct answer. For example, after getting the actual solution, with an evil mind, Bob may decide not to tell Alice the actual solution x. He can do this without being caught because he can send an arbitrary vector to Alice, who has no way to verify whether the received vector is the actual solution or not. This is not fair to Alice. To achieve the fairness, Alice should request Bob to send back a vector $v = M_2x - b_2$ along with the solution x. This vector does not give Alice any more power to derive Bob's data because if Bob is honest, Alice will know the value of $M_2x - b_2$ anyway because of $(M_1 + M_2)x = b_1 + b_2$. But if Bob still wants to cheat, he has to find two vectors x' and v', such that $M_1x' - b_1 = v'$. Without knowing M_1 and b_1, Bob cannot find these two vectors. The protocol is described in the following:

Protocol 1. *(PPC-LSE)* Alice has a matrix M_1 and a vector b_1, and Bob has a matrix M_2 and a vector b_2. M_1 and M_2 are $n \times n$ matrices; b_1 and b_2 are n-dimensional vector.

1. Bob generates two invertible random $n \times n$ matrices P and Q.
2. Alice and Bob use a secure protocol (will describe it later) to evaluate $M' = P(M_1 + M_2)Q$. Only Alice knows the result M'.
3. Alice and Bob use a secure protocol (will describe it later) to evaluate $b' = P(b_1 + b_2)$. Only Alice knows the result b'.
4. Alice solves the linear equations $M'\hat{x} = b'$. If the solution does not exist, Alice tells Bob so, then terminates the protocol. If the solution exists, Alice sends the solution $\hat{x}$ to Bob.
5. Bob computes $x = Q\hat{x}$ and $v = M_2x - b_2$, then sends both vectors x and v to Alice.
6. Alice checks whether x is the actual solution by verifying whether $||(M_1x - b_1) + v||$ equals to zero (or close to zero within the acceptable range if computation errors are inevitable).

Private Evaluation of $M' = P(M_1 + M_2)Q$

To privately evaluate M', Alice could send p matrices to Bob, with one of the matrices being M_1 and the rest of the matrices being random; however, Bob does not know which one is M_1. Then Bob computes the $P(H_i + M_2)Q$ for each matrices H_i he receives. At the end Alice uses the 1-out-of-N oblivious transfer protocol to get back from Bob one and only one of the result, the result of $M' = P(M_1 + M_2)Q$. Because of the way the 1-out-of-N oblivious transfer protocol works, Alice can decide which result to get, but Bob cannot learn which one Alice has chosen. However there is one drawback in this approach: if the value of M_1 has certain public-known properties, Bob might be able to differentiate M_1 from the other seemly random vectors. More seriously, after Bob finally gets the solution x, it only takes him p^2 tries to find both M_1 and b_1.

The above drawback can be fixed by dividing the matrix M_1 into m random matrices $X_1, \ldots, X_m$, with $M_1 = \sum_{i=1}^{m} X_i$. Alice and Bob can use the same method as described above to compute $P(X_i + M_2)Q$. As a result of the protocol, Alice gets $P(X_i + M_2)Q$ and Bob only knows one of the p vectors is X_i, but because of the randomness of X_i, Bob cannot find out which one is X_i. Certainly, there is 1 out p possibility that Bob could guess the correct X_i, but since M_1 is the sum of m such random matrices, the chance that Bob guess the correct M_1 is 1 out p^m, which could be very small if we chose p^m large enough.

However, knowing the values of $P(X_i + M_2)Q$ for $i = 1, \ldots, m$ might make it easier for Alice to figure out the value of M_2, therefore, Bob also needs to disguise the results of $P(X_i+M_2)Q$. One way to do this is to divide M_2 to m random matrices $(Y_1, \ldots, Y_m)$ as well, each time Bob returns the values of $P(X_i + Y_i)Q + R_i$ for $i = 1, \ldots, m$, where R_i's are also random matrices.

After Alice gets $P(X_i+Y_i)Q+R_i$ for $i = 1, \ldots, m$, she can sum them up and get $P(M_1 + M_2)Q + \sum_{i=1}^{m} R_i$. Bob can send the result of $\sum_{i=1}^{m} R_i$ to Alice who can then get $P(M_1 + M_2)Q$. Figure 2 explains how the protocol works. The detail of the protocol is described in the following:

Protocol 2. Alice has a Matrix M_1, and Bob has a Matrix M_2 and two random matrices P and Q.

1. Alice and Bob agree on two numbers p and m, such that p^m is so big that conducting p^m additions is computationally infeasible. For example, Alice and Bob could choose $p = 2$ and $m = 1024$.
2. Alice generates m random matrices $X_1, \ldots, X_m$, such that $M_1 = X_1 + \ldots + X_m$.
3. Bob generates m random matrices $Y_1, \ldots, Y_m$, such that $M_2 = Y_1 + \ldots + Y_m$.
4. For each $j = 1, \ldots, m$, Alice and Bob conduct the following sub-steps:
   (a) Alice sends the following sequence to Bob:
   $$(H_1, \ldots, H_p)$$
   where for a secret $1 \leq k \leq p$, $H_k = X_j$; the rest of the sequence are random matrices. k is

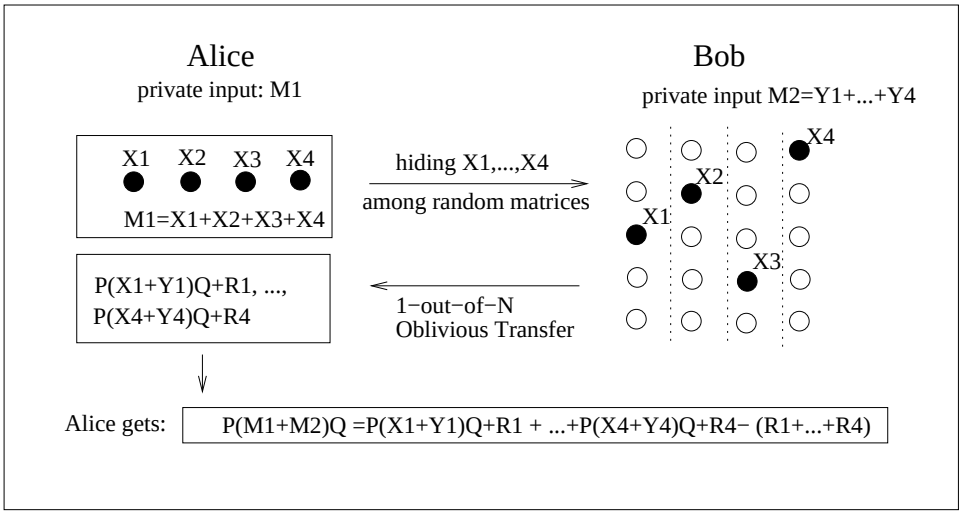

Figure 2. Private Evaluation of $P(M_1 + M_2)Q$

a secret random number known only by Alice, namely Bob does not know the position of X_j in the whole sequence.

(b) Bob computes $P(H_i + Y_j)Q + R_j$ for each $i = 1, \ldots, p$, where R_j is a random matrix.

(c) Using the 1-out-of-N Oblivious Transfer protocol, Alice gets back the result of

$$P(H_k + Y_j)Q + R_j = P(X_j + Y_j)Q + R_j$$

5. Bob sends $\sum_{j=1}^{m} R_j$ to Alice.

6. Alice computes $M' = \sum_{j=1}^{m}(P(X_j + Y_j)Q + R_j) - \sum_{j=1}^{m} R_j = P(M_1 + M_2)Q$.

Intuitively, Alice preserves her privacy by both dividing her matrix M_1 to p random matrices which are further hidden among many other random matrices, and by getting the results back using the 1-out-of-N oblivious transfer protocol. Bob's privacy is preserved by the 1-out-of-N oblivious transfer protocol, random matrices Y_i's and R_i's.

Theorem 1. *The protocol Π for computing $M' = P(M_1 + M_2)Q$ is private.*

Proof. We show a simulator S_1 for simulating $view_1^{\Pi}(M_1, M_2)$ such that $\{S_1(M_1, M'), -\}$ is indistinguishable from $\{(view_1^{\Pi}(M_1, M_2),\ output_2^{\Pi}(M_1, M_2))\}$. S_1 receives as input (M_1, M') (input/output) of Alice. Recall that the view of a party is defined by $(x, r, m_1, m_2, \ldots)$ where x is the input, r is the private coin tosses and m_i the ith message received.

- S_1, upon input (M_1, M') first chooses two invertible random matrices P' and Q' (these matrices simulate P and Q respectively).

- S_1 then finds M_2' (to simulate M_2) by solving $P'(M_1 + M_2')Q' = M'$.

- S_1 then generates m random matrices Y_i' for $i = 1, \ldots, m$, such that $\sum_{i=1}^{m} Y_i' = M_2'$.

- S_1 generates matrices X_i for $i = 1, \ldots, m$ using the same coin tosses r that Alice uses in generating these matrices.

- S_1 generates matrices R_i for $i = 1, \ldots, m$.

Let $S_1(M_1, M') = \{M_1, r, P'(X_1 + Y_1')Q' + R_1', \ldots, P'(X_m + Y_m')Q' + R_m', \sum_{i=1}^{n} R_i'\}$. Since $view_1^{\Pi}(M_1, M_2) = \{M_1, r,\ P(X_1 + Y_1)Q + R_1,\ \ldots,\ P(X_m + Y_m)Q + R_m, \sum_{i=1}^{n} R_i\}$. And $\{S_1(M_1, M'), -\}$ is computationally indistinguishable from $\{view_1^{\Pi}(M_1, M_2), -\}$

We now show a simulator S_2 for simulating $view_2^{\Pi}(M_1, M_2)$ such that $\{M', S_2(M_2, -)\}$ is indistinguishable from $\{(output_1^{\Pi}(M_1, M_2)), view_2^{\Pi}(M_1, M_2)\}$,

Bob generates $m * p$ random $n \times n$ matrices $\{(H_{1,1}', \ldots, H_{1,p}'),\ \ldots,\ (H_{m,1}', \ldots, H_{m,p}')\}$. Each element is uniformly distributed. Therefore, $S_2(M_2, -) = \{M_2, r,\ (H_{1,1}', \ldots, H_{1,p}'),\ \ldots,\ (H_{m,1}', \ldots, H_{m,p}')\}$. We also have $view_2^{\Pi}(M_1, M_2)\} = \{M_2, r,\ (H_{1,1}, \ldots, H_{1,p}),\ \ldots,\ (H_{m,1}, \ldots, H_{m,p})\}$. Because of the definition of $H_{i,j}$, $\{M', S_2(M_2, -)\}$ is computationally indistinguishable from $\{(output_1^{\Pi}(M_1, M_2)), view_2^{\Pi}(M_1, M_2)\}$.

□

Private Evaluation of $b' = P(b_1 + b_2)$

This protocol is similar to the protocol of evaluating M'. and the security property can be proved similarly.

Protocol 3. Alice has a vector b_1, Bob has a vector b_2 and a random matrix P.

1. Alice and Bob agree on two numbers p and m, such that p^m is so big that conducting p^m additions is computationally infeasible.
2. Alice generates m random vectors $x_1, \ldots, x_m$, such that $b_1 = x_1 + \ldots + x_m$.
3. Bob generates m random vectors $y_1, \ldots, y_m$, such that $b_2 = y_1 + \ldots + y_m$.
4. For each $j = 1, \ldots, m$, Alice and Bob conduct the following sub-steps:
   (a) Alice sends the following sequence to Bob:
   $$(h_1, \ldots, h_p)$$
   where for a secret $1 \leq k \leq p$, $h_k = x_j$; the rest of the sequence are random vectors. k is a secret random number known only by Alice, namely Bob does not know the position of x_j in the whole sequence.
   (b) Bob computes $P(h_i + y_j) + r_j$ for each $i = 1, \ldots, p$, where r_j is a random vector.
   (c) Using the 1-out-of-N Oblivious Transfer protocol, Alice gets back the result of
   $$P(h_i + y_j) + r_j = P(x_j + y_j) + r_j$$
5. Bob sends $\sum_{j=1}^{m} r_j$ to Alice.
6. Alice computes $b' = \sum_{j=1}^{m}(P(x_j + y_j) + r_j) - \sum_{j=1}^{m} r_j = P(b_1 + b_2)$.

Theorem 2. *The protocol for computing $b' = P(b_1 + b_2)$ is private.*

Theorem 3. *PPC-LSE protocol is a protocol for privately computing the solution to the Linear System of Equations problem.*

Proof. We need to show a simulator S_1 for simulating $view_1^{\Pi}((M_1, b_1), (M_2, b_2))$ such that $\{S_1((M_1, b_1), x), x\}$ is indistinguishable from $\{(view_1^{\Pi}((M_1, b_1), (M_2, b_1)), output_2^{\Pi}((M_1, b_1), (M_2, b_2)))\}$.

- Alice generates random matrix M', and then sets $b' = M'x$. M' is to simulate $P(M_1 + M_2)Q$, and b' is to simulate $P(b_1 + b_2)$.
- From the proof of the protocols for evaluating M' and b', we can similarly simulate Alice's view upon the input of (M_1, M') (resp., (b_1, b')).

Based on the proof of the protocols for evaluating M' and b', we know that $\{S_1((M_1, b_1), x), x\}$ is indistinguishable from $\{(view_1^{\Pi}((M_1, b_1), (M_2, b_2)), output_2^{\Pi}((M_1, b_1), (M_2, b_2)))\}$.

The design of the simulator S_2 is similarly based on the simulators used in the proof of the protocols for evaluating M' and b'.

□

3.3 Privacy-Preserving Cooperative Linear Least-Squares Problem

The linear system of equations problem consists of n equations of n unknown variables. There are situations where we have more equations to satisfy than the number of unknown variables. Most often, we cannot satisfy all of these equations, but we may find a solution that can satisfy them as best as we can. This problem is called the linear least-squares problem. We solve the privacy-preserving cooperative linear least-squares problem (PPC-LLS) in this subsection.

Problem 2. *(PPC-LLS)* Alice has a matrix M_1 and a vector b_1, and Bob has a matrix M_2 and a vector b_2, where M_1 and M_2 are $m \times n$ matrices ($m > n$), and b_1 and b_2 are m-dimensional vectors. Without disclosing their private inputs to the other party, Alice and Bob want to solve the linear equations

$$(M_1 + M_2)x = b_1 + b_2$$

Since there are more conditions (equations) to be satisfied than degrees of freedom (variables), it is unlikely that they can all be satisfied. Therefore, they want to attempt to satisfy the equations as best as they can–that is, make the size of the residual vector r with components

$$r_j = c_j - \sum_{i=1}^{n} a_{ji} x_i$$

as small as possible (a_{ji} are the entries in the new matrix $M = M_1 + M_2$, c_j are the entries in the new vector $b = b_1 + b_2$). The least-squares criterion is the use of the Euclidean (or least-squares) norm for the size of r; that is, minimize

$$\sqrt{\sum_{j=1}^{m} r_j^2} = ||r||_2$$

Solution: Linear least squares problem $Mx = b$ can be expressed in linear system:

$$M^T M x = M^T b$$

which contains n linear equations in the n unknowns x_i, hence can be solved using the usual methods for the linear equations problem, such as the the Gaussian elimination method and the Cholesky method, Such an approach to

solve the least-squares problem is called the normal equations approach because $M^T Mx = M^T b$ are normal equations.

In the privacy-preserving cooperative linear least-squares problem, $M = M_1 + M_2$, $b = b_1 + b_2$, therefore we have $M^T M = M_1^T M_1 + M_1^T M_2 + M_2^T M_1 + M_2^T M_2$, and $M^T b = M_1^T b_1 + M_1^T b_2 + M_2^T b_1 + M_2^T b_2$.

Therefore, the linear equations $M^T Mx = M^T b$ becomes the following:

$$(M_1^T M_1 + M_1^T M_2 + M_2^T M_1 + M_2^T M_2)x$$

$$= (M_1^T b_1 + M_1^T b_2 + M_2^T b_1 + M_2^T b_2)$$

Using the Matrix-Vector Product protocol and the Matrix Product protocol (both protocols will be described next), Alice and Bob can get the following:

$$V_1 + V_2 = M_1^T M_2$$

$$W_1 + W_2 = M_2^T M_1$$

$$v_1 + v_2 = M_1^T b_1$$

$$w_1 + w_2 = M_2^T b_2$$

where matrices V_1, W_1, vectors v_1 and w_1 are known only to Alice; matrices V_2, W_2, vectors v_2 and w_2 are known only to Bob. Let $M_1' = M_1^T M_1 + V_1 + W_1$, $M_2' = M_2^T M_2 + V_2 + W_2$, $b_1' = M_1^T b_1 + v_1 + w_1$, $b_2' = M_2^T b_2 + v_2 + w_2$, we have

$$(M_1' + M_2')x = b_1' + b_2'$$

where M_1' and M_2' are $n \times n$ matrices, and b_1' and b_2' are vectors of length n; M_1' and b_1' are known only to Alice, and M_2' and b_2' are known only to Bob. This is a PPC-LSE problem. It can be solved using the PPC-LSE protocol described in 3.2.

Protocol 4. *(Matrix Product Protocol)* Alice has a private matrix A, Bob has a private matrix B. At the end of the protocol, Alice gets R_a, and Bob gets R_b, where $R_a + R_b = AB$, R_a and R_b are random matrices.

1. Alice and Bob agree on two numbers p and m, such that p^m is so big that conducting p^m additions is computationally infeasible.

2. Alice generates m random matrices $X_1, \ldots, X_m$, such that $A = X_1 + \ldots + X_m$.

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

   (a) Alice sends the following sequence to Bob:

   $$(H_1, \ldots, H_p)$$

   where for a secret $1 \leq k \leq p$, $H_k = X_j$; the rest of the sequence are random matrices. k is a secret random number known only by Alice, namely Bob does not know the position of X_j in the whole sequence.

   (b) Bob computes $H_i B - R_j$ for each $i = 1, \ldots, p$, where R_j is a random matrix.

   (c) Using the 1-out-of-N Oblivious Transfer protocol, Alice gets back the result of

   $$H_k B - R_j = X_j B - R_j$$

4. Alice gets $R_a = \sum_{j=1}^{m} (X_j B - R_j) = AB - \sum_{j=1}^{m} R_j$, and Bob gets $R_b = \sum_{j=1}^{m} R_j$.

Protocol 5. *(Matrix-Vector Product Protocol)* Alice has a private matrix A, Bob has a private vector b. At the end of the protocol, Alice gets r_a, and Bob gets r_b, where $r_a + r_b = Ab$, R_a and R_b are random vectors.

The protocol is similar to the Matrix Product protocol. Just replace each occurrence of matrix B in the Matrix Product protocol with the vector b; replace the random matrix R_j with the random vector r_j for $j = 1, \ldots, m$; also replace the matrix R_a with the vector r_a, and R_b with r_b.

Protocol 6. *(PPC-LLS)*

1. Using the Matrix-Vector product protocol and the Matrix product protocol, Alice gets V_1, W_1, v_1, and w_1; Bob gets V_2, W_2, v_2, and w_2; where, U_i and W_i are matrices, v_i and w_i are vectors, and $V_1 + V_2 = M_1^T M_2$, $W_1 + W_2 = M_2^T M_1$, $v_1 + v_2 = M_1^T b_1$, $w_1 + w_2 = M_2^T b_2$.

2. Alice computes $M_1' = M_1^T M_1 + V_1 + W_1$ and $b_1' = M_1^T b_1 + v_1 + w_1$.

3. Bob computes $M_2' = M_2^T M_2 + V_2 + W_2$ and $b_2' = M_2^T b_2 + v_2 + w_2$.

4. Alice and Bob use PPC-LSE protocol to solve $(M_1' + M_2')x = b_1' + b_2'$.

The linear least-squares problem are normally used in regression and mathematical modeling. Consider building an investment model for a financial organization. One example is to model customers' investment as a function of age. In such a case the bank knows or believes or hopes there are n different factors–all related to the age–that influence the customers' decision on investment, and the bank wants to build a mathematical model according to these n factors. Formally speaking, the bank want to find out the function $b(t) = \sum_{i=1}^{n} x_i f_i(t)$, where t is the variable representing the age, and $f_i(t)$ express the different age factors.

Suppose now that the bank takes a large number of observation from the data it collected, and obtains values b_j

for t values t_j, $j = 1, \ldots, m$, and $m > n$. The problem of building such a mathematical model is just to solve the following linear least-square system:

$$d_j = \sum_{i=1}^{n} f_i(t_j)x_i, j = 1, \ldots, m$$

There are times when one financial organization does not have the sufficient data to build such a mathematical model, it thereby needs to cooperate with another financial organization, who also wants to benefit from such a cooperation. So both financial organizations would contribute their own data toward building such a model. Because this type of data usually consists of proprietary information that none of the financial organizations is willing to disclose to the others, these two financial organizations need to find a way to build the mathematical model without violating their privacy constraints. They can use PPC-LLS protocol.

Theorem 4. *PPC-LLS protocol is a protocol for privately computing the solution to the Linear Least-Squares Problem.*

The theorem is correct because the PPC-LLS protocol is reduced to the PPC-LSE protocol, which is already proved.

4 Protocol Efficiency

A Comparison to Generic Solutions.

The motivation of this research, i.e. designing specific solutions for each specific problems, is to reduce the communication cost. Therefore, in this section, we will compare the communication cost of our approach with that of the general solutions (the circuit evaluation approach)

For the PPC-LSE problem (and also for the PPC-LLE problem because it can be reduced to the PPC-LSE problem), assume the size of the matrix M is $n \times n$, and the d is the maximum length to represent a number in F. Assume that Gaussian elimination method is used in both the PPC-LSE protocol and the general solution.

As we know that the cost of Gaussian elimination takes $O(n^3)$ multiplication operations. And by a rough estimate, the size of a secure circuit for a single multiplication is about $O(d^2)$. Therefore, the total size of the circuit to conduct the Gaussian elimination is $O(n^3 * d^2)$.

In the PPC-LSE protocol, the cost of communication is $O(\mu * n^2)$, where μ is the security parameter. Since the difficulty to compromise the security is $O(2^\mu * n^2)$ (n^2 is introduced by the multiplication of a matrix and a vector, and 2^μ is introduced by the oblivious transfer), setting $\mu = 256$ is reasonably secure. Therefore the cost of communication $O(\mu * n^2)$ is significantly better than $O(n^3 * d^2)$.

5 Conclusion and Future Work

In this paper, we have defined a set of new privacy-preserving cooperative scientific computation problems: privacy-preserving cooperative linear system of equations problem and privacy-preserving cooperative linear least-square problem. We have developed protocols to solve these problems.

The major limitation of this work is due to the finite field assumption, which makes the computations in our paper somewhat different from the original scientific computations. In our future work, we would like to define a finite field that makes our computations consistent with the original scientific computations. Another alternative is to devise meaningful privacy requirements over infinite field, rather than using what Goldreich defined for a finite domain.

Rice points out that using $M^T Mx = M^T b$ to solve the linear least-square problem is not always the best approach, because it introduces the ill-conditioned matrix $M^T M$–the condition number of $M^T M$ is the condition number of M *squared* [20]. In the case where condition number of $M^T M$ is too bad, the solution might be random numbers unrelated to the original problem. In those cases, other approaches– such as the Gram-Schmidt Orthogonalization approach and the Orthogonal Matrix Factorization approach– are better than the normal equations approach. Developing protocols to solve the least-square problem using these approaches is an avenue we could pursue in the future work.

There are some other interesting scientific computation problems that we will study in the future work, such as how to compute *eigenvalues*, *eigenvectors*, *determinants*, *conditions*, and factorization of a matrix in the privacy-preserving cooperative computation situation.

Acknowledgments

We are very grateful to Rebecca Wright for her valuable suggestions and advice. We also thanks anonymous reviewers for their valuable comments.

References

[1] Rakesh Agrawal and Ramakrishnan 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] C. Cachin. Efficient private bidding and auctions with an oblivious third party. In *Proceedings of the 6th ACM conference on Computer and communications security*, pages 120–127, Singapore, November 1-4 1999.

[3] B. Chor and N. Gilboa. Computationally private information retrieval (extended abstract). In *Proceedings of the twenty-*

ninth annual ACM symposium on Theory of computing, El Paso, TX USA, May 4-6 1997.

[4] G. Brassard, C. Crepeau and J. Robert. All-or-nothing disclosure of secrets. In *Advances in Cryptology - Crypto86, Lecture Notes in Computer Science*, volume 234-238, 1987.

[5] Y. Desmedt. Some recent research aspects of threshold cryptography. In *Lecture Notes in Computer Science 1396*, pages 158–173. Springer-Verlag, 1997.

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

[7] S. Evan, O. Goldreich and A. Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28:637–647, 1985.

[8] S. Goldwasser. Multi-party computations: Past and present. In *Proceedings of the sixteenth annual ACM symposium on Principles of distributed computing*, Santa Barbara, CA USA, August 21-24 1997.

[9] 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.

[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] Y. Ishai and E. Kushilevitz. Improved upper bounds on information-theoretic private information retrieval (extended abstract). In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, Atlanta, GA USA, May 1-4 1999.

[12] 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.

[13] 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.

[14] Y. Gertner, Y. Ishai, E. Kushilevitz and T. Malkin. Protecting data privacy in private information retrieval schemes. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, Dallas, TX USA, May 24-26 1998.

[15] C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems*. Prentice-Hall, Englewood Cliffs, 1974.

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

[17] 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.

[18] 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.

[19] 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.

[20] J. R. Rice. *Matrix Computations and Mathematical Software*. McGraw-Hill Book Company, 1981.

[21] A. Shamir. How to share a secret. *Communication of the ACM*, 22(11):612–613, 1979.

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