Privacy-Preserving Cooperative Statistical Analysis *†

Wenliang Du
Center for Systems Assurance
Department of Electrical Engineering and Computer Science
Syracuse University, 121 Link Hall, Syracuse, IN 13244
Email: wedu@ecs.syr.edu Tel: (315)-443-9180

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: mja@cs.purdue.edu

Abstract

The growth of the Internet opens up tremendous opportunities for cooperative computation, where the answer depends on the private inputs of separate entities. Sometimes these computations may occur between mutually untrusted entities. The problem is trivial if the context allows the conduct of these computations by a trusted entity that would know the inputs from all the participants; however if the context disallows this then the techniques of secure multi-party computation become very relevant and can provide useful solutions.

Statistic analysis is a widely used computation in real life, but the known methods usually require one to know the whole data set; little work has been conducted to investigate how statistical analysis could be performed in a cooperative environment, where the participants want to conduct statistical analysis on the joint data set, but each participant is concerned about the confidentiality of its own data. In this paper we have developed protocols for conducting the statistic analysis in such kind of cooperative environment based on a data perturbation technique and cryptography primitives.

*Portions of this work were supported by Grant EIA-9903545 from the National Science Foundation, by sponsors of the Center for Education and Research in Information Assurance and Security at Purdue University, and by the Center for Computer Application and Software Engineering (CASE) at Syracuse University.

†In Proceedings of the *17th Annual Computer Security Applications Conference*, pages 102-110, New Orleans, Louisiana, USA, December 10-14, 2001.

1 Introduction

The growth of the Internet opens up tremendous opportunities for cooperative computation, where the answer depends on the private inputs of separate entities. Sometimes these computations may occur between mutually untrusted entities. The problem is trivial if the context allows the conduct of these computations by a trusted entity that would know the inputs from all the participants; however if the context disallows this then the techniques of secure multi-party computation become very relevant and can provide useful solutions.

In this paper we investigate how various statistical analysis problems could be solved in a cooperative environment, where two parties need to conduct statistical analysis on the joint data set. We call the new problems *secure two-party* statistical analysis problems.

Basic statistic analysis operations consist of computing the mean value of a data set, the standard deviation, the correlation coefficient between two different features, the regression line and so on. If one knows the full data set, one can use the standard equations described in most of the fundamental statistics books to conduct the analysis. However, in the cooperative environment, one might need to conduct statistical analysis without being able to know the full data set because of the privacy constraints. The following examples illustrate this kind of situation:

- A school wants to investigate the relationship between people's intelligence quotient (IQ) scores and their annual salaries. The school has its students' IQ scores, but does not have students'

salary information; therefore the school needs to cooperate with companies that hire the students, but those companies are not willing to disclose the salary information. On the other hand, the school cannot give students' IQ scores to their employers either.

- Two retail companies A and B each have a data set about their own customers' buying behaviors. Both companies want to conduct statistical analysis on their joint data set for mutual benefit. Since these two companies are competitors in the market, they do not want to disclose the detailed customers' information to the other company, but they feel comfortable disclosing only the aggregate information.

The standard statistical analysis methods cannot easily extend to solve the above problems; we need methods that support statistical analysis in a privacy-preserving manner. The goal of this paper is to develop protocols for this type of cooperative statistical analysis.

There are two common ways of cooperation in practice. For example, suppose X and Y are two different features of a sample, and they are both used for a statistical analysis. In a cooperative environment, sometimes both cooperating parties can observe both X and Y features of the sample, while at some other time, one party can only observe X feature of the sample, and the other party can only observe Y feature of the same sample. The difficulties of cooperation in these situations are different. Therefore, based on whether the two cooperating parties could observe the same features of a sample or not, we formalized two different models for the secure two-party statistical analysis: the heterogeneous model and the homogeneous model. As we will show later, the solutions to these two different models are very different.

To conduct the statistical analysis in a cooperative environment, data exchange between the two parties is needed. However, to preserve the privacy of the data, no party should send its data to the other party in plain. In our solution, we use a data perturbation technique, namely we add random numbers to the original data to disguise the original data. Conducting statistical analysis based on the perturbed data surely produces a perturbed and therefore wrong result. In this paper we have demonstrated various ways to remove, without compromising the privacy, the perturbation from the result to produce a correct result. Our techniques are based on several cryptography primitives, such as 1-out-of-n Oblivious Transfer protocol and homomorphic encryption schemes.

Most of the statistical analysis computation investigated in this paper involve scalar product of two private vectors, each of which comes from a different party; therefore, we have studied the private scalar product problem independently, and use the solution to build the protocols for the statistical analysis problem. In addition to being used in this paper, the private scalar product protocol could also be used to solve many other secure two-party problems, such as secure two-party computational geometry problems discussed in [1]. We will discuss the private scalar product problem independently in the paper.

In this preliminary study, we assume that all parties are semi-honest; informally speaking, a semi-honest party is one who follows the protocol properly with the exception that it keeps a record of all its intermediate computations and might try to derive other parties' private inputs from the record. This semi-honest model is one of the widely adopted models in the studies of general secure multi-party computation problem.

Section 2 describes related work and some cryptography primitives will be used in this paper. Section 3 presents an important building block, the scalar product protocol, which will be used later to solve secure two-party statistical analysis problems. Section 4 presents the definitions of secure two-party statistical analysis problems and their solutions. Finally section 5 concludes this paper and proposes several future research directions.

2 Related Work

Secure Multi-Party Computation

The secure two-party statistical analysis problems described in the previous section are actually special cases of the general Secure Multi-party Computation problem [13, 8, 5]. Generally speaking, a secure multi-party computation problem deals with computing a 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. The history of the multi-party computation problem is extensive since it was introduced by Yao [13] and extended by Goldreich, Micali, and Wigderson [8], and by many others.

In theory, the general secure multi-party computation problem is solvable using circuit evaluation protocol [13, 8, 5]. In the circuit evaluation protocol, each functionality F is represented as a Boolean circuit, and then the parties run a protocol for every gate in the circuit. While this approach is appealing in its gener-

ality, the communication complexity of the protocol it generates depends on the size of the circuit that expresses the functionality F to be computed, and in addition, involves large constant factors in their complexity. Therefore, as Goldreich points out in [5], 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. This is our motivation of seeking special solutions to statistical analysis problems, solutions that are more efficient than the general theoretical solutions.

1-out-of-N Oblivious Transfer

Goldreich's circuit evaluation protocol uses the 1-out-of-N Oblivious Transfer, and our protocols in this paper also heavily depends on this protocol. An 1-out-of-N Oblivious Transfer protocol [6, 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 [10] by Naor and Pinkas. By combining this protocol with the scheme by Cachin, Micali and Stadler [7], the 1-out-of-N Oblivious Transfer protocol could be achieved with polylogarithmic (in n) communication complexity.

Homomorphic Encryption Schemes

We need a public-key cryptosystems with a *homomorphic* property for some of our protocols: $E_k(x) * E_k(y) = E_k(x + y)$. Many such systems exist, and examples include the systems by Benaloh [3], Naccache and Stern [9], Okamoto and Uchiyama [11], Paillier [12], to mention a few. A useful property of homomorphic encryption schemes is that an "addition" operation can be conducted based on the encrypted data without decrypting them.

3 New Building Blocks

In this section, we introduce a secure two-party protocols: the scalar product protocol. This protocol serves as an important building block in solving the secure two-party statistical analysis problems considered later in the paper. This protocol is first presented in [1].

3.1 Scalar Product Protocol

We use $X \cdot Y$ to denote the scalar product of two vectors $X = (x_1, \ldots, x_n)$ and $Y = (y_1, \ldots, y_n)$, $X \cdot Y = \sum_{k=1}^{n} x_k y_k$. Our definition of the problem is slightly different and more general: We assume that Alice has the vector X and Bob has the vector Y, and the goal of the protocol is for Alice (but not Bob) to get $X \cdot Y + v$ where v is random and known to Bob only (of course without either side revealing to the other the private data they start with). Our protocols can easily be modified to work for the version of the problem where the random v is given ahead of time as part of Bob's data (the special case $v = 0$ puts us back in the usual scalar product definition). 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 the multiple-step protocol, the effect of v can be effectively "subtracted out" by Bob without revealing v to Alice (this should become clearer with example protocols that we later give).

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.

We have developed two protocols, and we will present both of them here.

3.1.1 Scalar Product Protocol 1

Consider the following naive solution: Alice sends p vectors to Bob, only one of which is X (the others are arbitrary). Then Bob computes the scalar products between Y and each of these p vectors. At the end Alice uses the 1-out-of-p oblivious transfer protocol to get back from Bob the product of X and Y. Because of the way oblivious transfer protocol works, Alice can decide which scalar product to get, but Bob could not learn which one Alice has chosen. There are many drawbacks to this approach: If the value of X has certain public-known properties, Bob might be able to differentiate X from the other $p - 1$ vectors, but even if Bob is unable to recognize X his chances of guessing it is 1 out of p, unacceptable in many situations.

The above drawbacks can be fixed by dividing vector X into m random vectors $V_1, \ldots, V_m$ of which it is the sum, i.e., $X = \sum_{i=1}^{m} V_i$. Alice and Bob can use the above naive method to compute $V_i \cdot Y + r_i$, where r_i is a random number and $\sum_{i=1}^{n} r_i = v$ (see Figure 1).

As a result of the protocol, Alice gets $V_i \cdot Y + r_i$ for $i = 1, \ldots, m$. Because of the randomness of V_i and its position, Bob could not find out which one is V_i. Certainly, there is 1 out p possibility that Bob can guess the correct V_i, but since X is the sum of m such random vectors, the chance that Bob guesses the correct X is 1 out p^m, which could be very small if we chose p^m large enough.

After Alice gets $V_i \cdot Y + r_i$ for $i = 1, \ldots, n$, she can compute $\sum_{i=1}^{m}(V_i \cdot Y + r_i) = X \cdot Y + v$. The detailed protocol is described in the following:

Protocol 1. *(Two-Party Scalar Product Protocol 1)*
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 $X \cdot Y + v$ where v is a random scalar known to Bob only.

1. Alice and Bob agree on two numbers p and m, such that p^m is large enough.
2. Alice generates m random vectors, $V_1, \ldots, V_m$, such that $X = \sum_{j=1}^{m} V_i$.
3. Bob generates m random numbers $r_1, \ldots, r_m$ such that $v = \sum_{j=1}^{m} r_j$.
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 $H_k = V_j$, and the rest of H_i's are random vectors. Because k is a secret number known only to Alice, Bob does not know the position of V_j.
   (c) Bob computes $Z_{j,i} = H_i \cdot Y + r_j$ for $i = 1, \ldots, p$.
   (d) Using the 1-out-of-p Oblivious Transfer protocol, Alice gets $Z_j = Z_{j,k} = V_j \cdot Y + r_j$, while Bob learns nothing about k.
5. Alice computes $u = \sum_{j=1}^{m} Z_j = X \cdot Y + v$.

How is privacy achieved:

- If Bob chooses to guess, his chance of guessing the correct X is 1 out of p^m.
- The purpose of r_j is to add randomness to $V_j \cdot Y$, thus preventing Alice from deriving information about Y.

The communication complexity of the above protocols is $O(mp)$. We can improve it to $O(m + p)$ by using the following scheme: Alice sends $V_1, \ldots, V_m$ and $H_1, \ldots, H_p$ altogether to Bob then doing m-out-of-$(m + p)$ oblivious transfer. The probability of Bob guessing correct X is now 1 out $C(m, m + p)$, which could be small enough if we choose an appropriate value for p.

3.1.2 Scalar Product Protocol 2

Our next solution does not rely on 1-out-of-n Oblivious Transfer cryptography primitive as the previous one does, but is instead based on a homomorphic public key system. In the following discussion, we define $\pi(X)$ as another vector whose elements are random permutation of those of vector X.

We begin with two observations. First, a property of the scalar product $X \cdot Y$ is that $\pi(X) \cdot \pi(Y) = X \cdot Y$, regardless of what π is. Secondly, if Bob sends a vector $\pi(V)$ to Alice, where π and V are known only to Bob, Alice's chance of guessing the position of any single element of the vector V is 1 out of n (n is the size of the vector); Alice's chance of guessing the positions of all of the elements of the vector V is 1 out of $n!$.

A naive solution would be to let Alice get both $\pi(X)$ and $\pi(Y)$ but not π. Let us ignore for the time being the drawback that Alice gets the items of Y in permuted order, and let us worry about not revealing π to Alice: Letting Alice know $\pi(X)$ allows her to easily figure out the permutation function π from knowing both X and $\pi(X)$. In order to avoid this problem, we want to let Alice know only $\pi(X + R_b)$ instead of $\pi(X)$, where R_b is a random vector known only to Bob. Because of the randomness of $X + R_b$, to guess the correct π, Alice's chance is only 1 out of $n!$. Therefore to get the final scalar product, Bob only needs to send $\pi(Y)$ and the result of $R_b \cdot Y$ to Alice, who can compute the result of the scalar product by using

$$X \cdot Y = \pi(X + R_b) \cdot \pi(Y) - R_b \cdot Y$$

Now we turn our attention to the drawback that giving Alice $\pi(Y)$ reveals too much about Y (for example, if Alice is only interested in a single element of the vector Y, her chance of guessing the right one is an unacceptably low 1 out of n). One way to fix this is to divide Y to m random pieces, $V_1, \ldots, V_m$, with $Y = V_1 + \ldots + V_m$; then Bob generates π random permutations $\pi_1, \ldots, \pi_m$ (one for each "piece" V_i of Y) and lets Alice know $\pi_i(V_i)$ and $\pi_i(X + R_b)$ for $i = 1, \ldots, m$. Now in order to guess the correct value of a single element of Y, Alice has to guess the correct position of V_i in each one of the m rounds; the possibility of a successful guessing becomes 1 out of n^m.

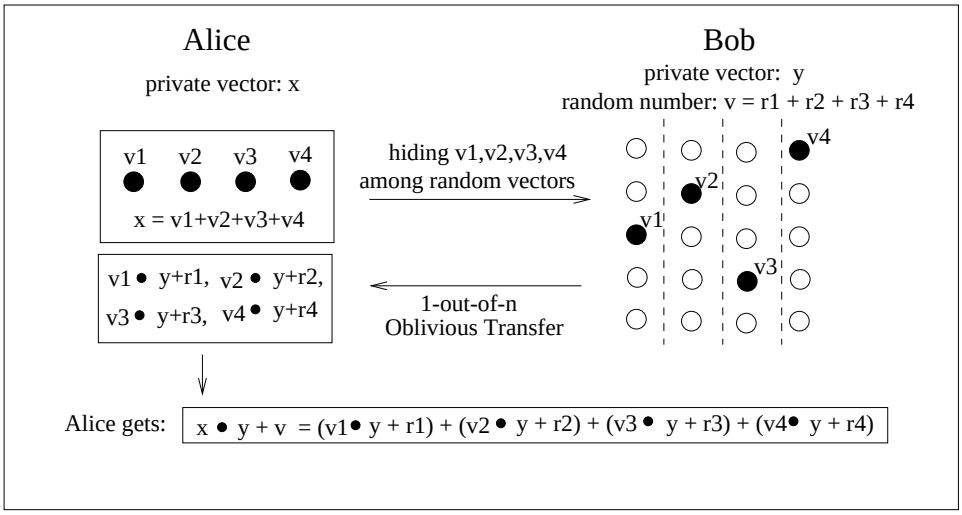

Figure 1. Scalar Product Protocol 1

Now, let us consider the unanswered question: how could Alice get $\pi(X + R_b)$ without learning π or R_b? We do this with a technique based on a homomorphic public key system, that was used in [2] in a different context (to compute the minimum value in a vector that is the difference of Alice's private vector and Bob's private vector). Recall that an encryption scheme is *homomorphic* if $E_k(x) * E_k(y) = E_k(x+y)$. A good property of homomorphic encryption schemes is that "addition" operation can be conducted based on the encrypted data without decrypting them. Based on the homomorphic public key system, we have the following Permutation Protocol (where, for a vector $Z = (z_1, \ldots, z_n)$, we define $E(Z) = (E(z_1), \ldots, E(z_n))$, $D(Z) = (D(z_1), \ldots, D(z_n))$):

Protocol 2. *(Permutation Protocol)*

Inputs: Alice has a vector X. Bob has a permutation π and a vector R.

Output: Alice gets $\pi(X + R)$.

1. Alice generates a key pair for a homomorphic public key system and sends the public key to Bob. The corresponding encryption and decryption is denoted as $E(\cdot)$ and $D(\cdot)$.

2. Alice encrypts $X = (x_1, \ldots, x_n)$ using her public key and sends $E(X) = (E(x_1), \ldots, E(x_n))$ to Bob.

3. Bob computes $E(R)$, then computes $E(X) * E(R) = E(X + R)$; Bob then permutes $E(X + R)$ using the random permutation function π, thus getting $\pi(E(X + R))$; Bob sends the result of $\pi(E(X + R))$ to Alice.

4. Alice computes $D(\pi(E(X + R))) = \pi(D(E(X + R))) = \pi(X + R)$.

Based on Secure Two-Party Permutation Protocol, we have developed the following scalar product protocol:

Protocol 3. *(Secure Two-Party Scalar Product Protocol 2)*

Inputs: Alice has a secret vector X, Bob has a secret vector Y.

Output: Alice gets $X \cdot Y + v$ where v is a random scalar known to Bob only.

1. Bob's set up:

   (a) Bob divides Y to m random pieces, s.t. $Y = V_1 + \ldots + V_m$.

   (b) Bob generates m random vectors $R_1, \ldots, R_m$, let $v = \sum_{i=1}^{m} V_i \cdot R_i$.

   (c) Bob generates m random permutations $\pi_1, \ldots, \pi_m$.

2. For each $i = 1, ..., m$, Alice and Bob do the following:

   (a) Using Secure Two-Party Permutation Protocol, Alice gets $\pi_i(X + R_i)$ without learning either π_i or R_i.

   (b) Bob sends $\pi_i(V_i)$ to Alice.

   (c) Alice computes $Z_i = \pi_i(V_i) \cdot \pi_i(X + R_i) = V_i \cdot X + V_i \cdot R_i$

3. Alice computes $u = \sum_{i=1}^{m} Z_i = \sum_{i=1}^{m} V_i \cdot X + \sum_{i=1}^{m} V_i \cdot R_i = X \cdot Y + v$

How is privacy achieved:

- The purpose of R_i is to prevent Alice from learning π_i.
- The purpose of π_i is to prevent Alice from learning V_i. Although Alice learns a random permutation of the V_i, she does not learn more because of the randomness of V_i. Without π_i, Alice could learn each single value of V_i.
- If Alice chooses to guess, in order to successfully guess all of the elements in Y, her chance is $(\frac{1}{n!})^m$.
- Alice's chance of successfully guessing just one elements of Y is 1 out of n^m. For example, in order to guess the kth element of Y, Alice has to guess the the corresponding elements in $\pi_i(V_i)$ for all $i = 1, \ldots, m$. Because for each single i, the possibility is 1 out of n, the total possibility is 1 out of n^m.
- A drawback of this protocol is that the information about $\sum_{i=1}^{n} y_i$ is disclosed because the random permutation does not help to hide this information.

3.1.3 Implementation Issues

During the implementation, we need to consider the padding issues because most of the encryption scheme require padding if the size of a number is smaller than the expected size. For the security reason, A fixed padding cannot be used because it makes brute force attack possible. However, if random padding is used, how could Alice in Protocol 2 get the value of $x + y$ (x is Alice's number and y is Bob's number) without knowing how Bob pads his number y? We describe a padding scheme in the following:

Let p be the required size of a block for the encryption, and $|x| \leq \frac{1}{3}p$ and $|y| \leq \frac{1}{3}p$. When encrypting x, the encrypter randomly chooses a number r_1 such that $|r_1| = \frac{2}{3}p - 3$. The encryption is conducted on $0x00r_1$. When encrypting y, we choose a number r_2 such that $|r_2| = \frac{2}{3}p - 3$. The encryption is conducted on $0y00r_2$. In this way the encryption is a randomized one which can resist brute force searching. On the other hand the homomorphic property is conditionally guaranteed, because we have $E(0x00r_1)E(0y00r_2) = E((x + y)0(r_1 + r_2))$ and $x + y$ can be easily obtained from the decryption without knowing either r_1 or r_2. Although this scheme does not have the property of $E(x_1) \ldots E(x_n) = E(x_1 + \ldots + x_n)$, it does not affect our protocols.

3.1.4 Complexity Analysis

In the following discussion, we assume that d is the number of bits needed to represent any number in the inputs,

The communication cost of Protocol 3 is $4m * n * d$, where m is a security parameter (so that $\mu' = n^m$ is large enough). The communication cost of Protocol 1 is $p * t * n * d$, where $p \geq 2$ and t are security parameters such that $\mu'' = p^t$ is large enough. Setting $\mu' = \mu'' = \mu$ for the sake of comparison, the communication cost of Protocol 3 is $4 \log \mu \frac{nd}{\log n}$ and the communication cost of Protocol 1 is $\frac{p \log \mu}{\log p} nd$. When n is large, Protocol 3 is more efficient than Protocol 1.

The communication cost of the circuit evaluation protocol is $c * n * d^2$, where c is the number of bits sent over the network in the 1-out-of-n Oblivious Transfer protocol. Although the value of c depends on the specific implementation of the protocol, it is reasonable to assume $c = d$; therefore the communication cost becomes $n * d^3$, which is significantly more expensive than our scalar product protocols.

4 Secure Two-Party Statistical Analysis Problems and Protocols

4.1 Statistical Analysis Background

Without loss of generality, throughout this paper, we will use a data set D of size n that only consists of two different features x and y, where $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$.

As a preliminary study on the topic of secure two-party statistical analysis, we only focus on several basic statistical analysis, which are reviewed in the the following:

- Mean Value: $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$.
- Correlation Coefficient between x and y: Correlation coefficient measures the strength of a linear relationship between x and y, namely the degree to which larger x values go with larger y values and smaller x values go with smaller y values. Correlation coefficient r is computed using the following equation:

$$\begin{aligned} r &= \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}} \\ &= \frac{\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}}{\sqrt{(\sum_{i=1}^{n} x_i^2 - n\bar{x}^2)(\sum_{i=1}^{n} y_i^2 - n\bar{y}^2)}} \end{aligned}$$

- Linear Regression Line: The purpose of linear regression is to find the line that comes closest to your data. More precisely, the linear regression program finds values for the slope and intercept that define the line that minimizes the sum of the square of the vertical distances between the points and the line. The linear regression line is represented by the following equation: $y = bx+(\bar{y}-b\bar{x})$, where

$$b = \frac{\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}}{\sum_{i=1}^{n} x_i^2 - n\bar{x}^2}$$

4.2 Two Models of Cooperation

There are many ways two parties could cooperate in performing statistical analysis; Figure 2 describes two ways of cooperation that are common in practice. The first one is the heterogeneous cooperation model (Figure 2.b). In this model, each party holds different features of a data set. For example, if the whole data set consists of employees' salaries and ages, in a heterogeneous model, Alice could hold the salary information while Bob holds the age information.

The second way of cooperation is the homogeneous cooperation model (Figure 2.c). In this model, both party hold the same features, but each party holds a different subset of the data set. For instance, in a homogeneous model, Alice could hold department A's employee information while Bob holds department B's employee information.

Both of the above cooperation models are quite common in practice. In this paper, we have formally defined secure two-party statistical analysis problems corresponding to these cooperation models, and have developed protocols for those problems.

4.3 Heterogeneous Model

Problem 2. *(Secure Two-Party Statistical Analysis Problem in Heterogeneous Model)* Alice has a data set $D_1 = (x_1, \ldots, x_n)$, and Bob has another data set $D_2 = (y_1, \ldots, y_n)$, where x_i is the value of variable x, and y_i is the corresponding value of variable y. Alice and Bob want to find out the following:

1. *correlation coefficient* r between x and y.
2. *regression line* $y = bx + (\bar{y} - b\bar{x})$.

Correlation Coefficient Let $u = \sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}$, and $v = \sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}$. To compute the correlation coefficient r, we have the following equations:

$$\begin{aligned} r &= \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}} \\ &= \sum_{i=1}^{n} \frac{(x_i - \bar{x})}{u} \frac{(y_i - \bar{y})}{v} \\ &= \left(\frac{x_1 - \bar{x}}{u}, \ldots, \frac{x_n - \bar{x}}{u}\right) \cdot \left(\frac{y_1 - \bar{y}}{v}, \ldots, \frac{y_n - \bar{y}}{v}\right) \end{aligned}$$

This indicates that the task of computing the correlation coefficient is reduced to a secure two-party scalar product problem. It can be computed using Scalar Product Protocol (Protocol 1 or 3).

Linear Regression Line Let $w = \sum_{i=1}^{n} x_i^2 - n\bar{x}^2$. Because computing w only requires the value of variable x, it can be calculated by Alice alone. Therefore, we can use the following equations to compute the slope of the linear regression line:

$$\begin{aligned} b &= \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} x_i^2 - n\bar{x}^2} \\ &= \left(\frac{x_1 - \bar{x}}{w}, \ldots, \frac{x_n - \bar{x}}{w}\right) \cdot (y_1 - \bar{y}, \ldots, y_n - \bar{y}) \end{aligned}$$

This indicates that the task of computing b is also reduced to a secure two-party scalar product problem, and thus can be solved using Scalar Product Protocol (Protocol 1 or 3). The details of the protocol are described in the following:

Protocol 4. *(Secure Two-Party Statistical Analysis Protocol in Heterogeneous Model))*
Inputs: Alice has a data set $D_1 = (x_1, \ldots, x_n)$, and Bob has another data set $D_2 = (y_1, \ldots, y_n)$.
Outputs: Alice and Bob gets r and b.

1. Alice computes $\bar{x}$, $u = \sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}$, and $w = \sum_{i=1}^{n} x_i^2 - n\bar{x}^2$.
2. Bob computes $\bar{y}$ and $v = \sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}$.
3. Alice and Bob use Scalar Product Protocol (Protocol 1 or 3) to compute

$$\begin{aligned} r &= \left(\frac{x_1 - \bar{x}}{u}, \ldots, \frac{x_n - \bar{x}}{u}\right) \cdot \left(\frac{y_1 - \bar{y}}{v}, \ldots, \frac{y_n - \bar{y}}{v}\right) \\ b &= \left(\frac{x_1 - \bar{x}}{w}, \ldots, \frac{x_n - \bar{x}}{w}\right) \cdot (y_1 - \bar{y}, \ldots, y_n - \bar{y}) \end{aligned}$$

4.4 Homogeneous Model

Problem 3. *(Secure Two-Party Statistical Analysis Problem in Homogeneous Model)* Alice has a data set $D_1 = ((x_1, y_1), \ldots, (x_k, y_k))$, and Bob has another data

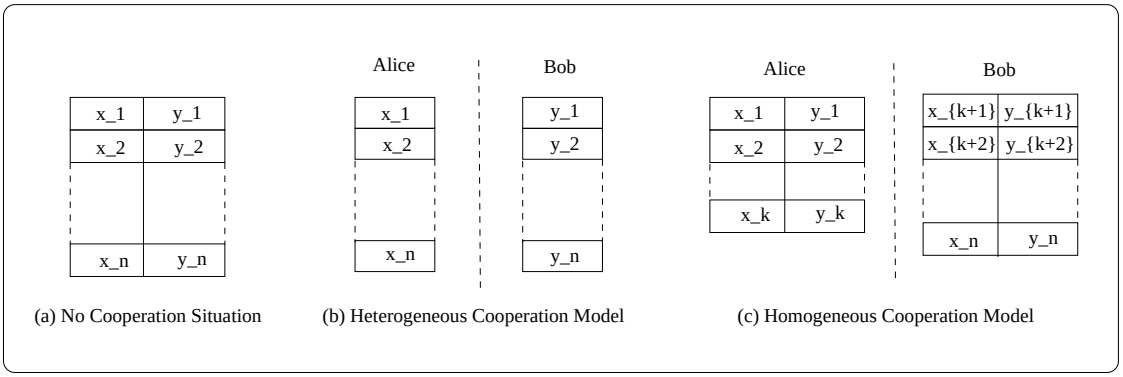

Figure 2. Two Models of Cooperation

set $D_2 = ((x_{k+1}, y_{k+1}), \ldots, (x_n, y_n))$, where x_i is the value of variable x, and y_i is the corresponding value of variable y. Alice and Bob want to find out the following:

1. *mean value* $\bar{x}$ (resp., $\bar{y}$).
2. *correlation coefficient* r between x and y.
3. *regression line* $y = bx + (\bar{y} - b\bar{x})$.

Let us first consider the above problem under the following privacy constraint:

> **Privacy Constraint A:** Alice does not want to disclose the information about D_1 other than the aggregate information including $\sum_{i=1}^{k} x_i$, $\sum_{i=1}^{k} x_i^2$, $\sum_{i=1}^{k} y_i$, $\sum_{i=1}^{k} y_i^2$, and $\sum_{i=1}^{k} x_i y_i$. Accordingly, Bob does not want to disclose the information about D_2 other than the aggregate information including $\sum_{i=k+1}^{n} x_i$, $\sum_{i=k+1}^{n} x_i^2$, $\sum_{i=k+1}^{n} y_i$, $\sum_{i=k+1}^{n} y_i^2$, and $\sum_{i=k+1}^{n} x_i y_i$.

Under Privacy Constraint A, computing mean value is trivial because both parties know $\sum_{i=1}^{n} x_i$ and $\sum_{i=1}^{n} y_i$. After getting $\bar{x}$ and $\bar{y}$, computing the correlation coefficient and the linear regression line is straightforward according to the following equations:

$$r = \frac{(\sum_{i=1}^{k} x_i \cdot y_i + \sum_{i=k+1}^{n} x_i \cdot y_i) - n * \bar{x}\bar{y}}{\sqrt{(\sum_{i=1}^{k} x_i^2 + \sum_{i=k+1}^{n} x_i^2) - n\bar{x}^2}} \cdot \frac{1}{\sqrt{(\sum_{i=1}^{k} y_i^2 + \sum_{i=k+1}^{n} y_i^2) - n\bar{y}^2}}$$

$$b = \frac{(\sum_{i=1}^{k} x_i \cdot y_i + \sum_{i=k+1}^{n} x_i \cdot y_i) - n * \bar{x}\bar{y}}{(\sum_{i=1}^{k} x_i^2 + \sum_{i=k+1}^{n} x_i^2) - n\bar{x}^2}$$

Now let us consider the same problem under a more strict privacy constraint:

> **Privacy Constraint B:** Alice and Bob do not want to disclose too much information about their data; more specifically, they do not want to disclose any more information than what can be derived from $\bar{x}$, $\bar{y}$, r and b. This implies that Alice can disclose $\sum_{i=1}^{k} x_i$ and $\sum_{i=1}^{k} y_i$ to Bob, and Bob can disclose $\sum_{i=k+1}^{n} x_i$ and $\sum_{i=k+1}^{n} y_i$ to Alice because those can be derived from $\bar{x}$ and $\bar{y}$.

Under this privacy constraint, computing the mean value is still trivial, but computing the correlation coefficient r and the linear regression line is not. In what follows, we demonstrate how to compute r (the linear regression line can be computed similarly).

Let $a_1 = \sum_{i=1}^{k} x_i \cdot y_i - k\bar{x}\bar{y}$, $b_1 = \sum_{i=k+1}^{n} x_i \cdot y_i - (n-k)\bar{x}\bar{y}$, $a_2 = \sum_{i=1}^{k} x_i^2 - k\bar{x}^2$, $b_2 = \sum_{i=k+1}^{n} x_i^2 - (n-k)\bar{x}^2$, $a_3 = \sum_{i=1}^{k} y_i^2 - k\bar{y}^2$, and $b_3 = \sum_{i=k+1}^{n} y_i^2 - (n-k)\bar{y}^2$. Note that a_i is only known to Alice, and b_i is only known to Bob. We have

$$r^2 = \frac{(a_1 + b_1)^2}{(a_2 + b_2)(a_3 + b_3)} = \frac{(a_1^2 + 2a_1b_1 + b_1^2)}{(a_2a_3 + b_2a_3 + a_2b_3 + b_2b_3)}$$

By using Scalar Product Protocol, we can let Alice learn u_1 and u_2, and let Bob learn v_1 and v_2, where $u_1 + v_1 = a_1^2 + 2a_1b_1 + b_1^2$ and $u_2 + v_2 = a_2a_3 + b_2a_3 + a_2b_3 + b_2b_3$. Now the question becomes how to compute $\frac{u_1+v_1}{u_2+v_2}$.

Problem 4. *(Division Problem)* Alice has u_1 and u_2; Bob has v_1 and v_2. Alice and Bob want to compute $z = \frac{u_1+v_1}{u_2+v_2}$. Alice should not learn v_1 or v_2; Bob should not learn u_1 or u_2.

In the following protocol, we first let Bob generate two random numbers r_1 and r_2; then we let Alice (only Alice) get the result of $z_1 = r_1(u_1 + v_1)$,

$z_2 = r_2(u_2 + v_2)$, and $r = \frac{r_2}{r_1}$. Therefore, Alice can compute $z = \frac{rz_1}{z_2} = \frac{u_1+v_1}{u_2+v_2}$. If r_1 and r_2 are both real numbers, Alice could not learn v_1 (resp., v_2) from z_1 (resp., z_2).

Protocol 5. *(Division Protocol)*
Input: Alice has u_1 and u_2; Bob has v_1 and v_2.
Output: Alice and Bob both gets the result of $z = \frac{u_1+v_1}{u_2+v_2}$

1. Bob generates two random numbers r_1 and r_2, and sends $r = \frac{r_2}{r_1}$ to Alice.
2. Alice and Bob use Scalar Product Protocol on $(u_1, 1)$ and (r_1, r_1v_1) to get $z_1 = r_1(u_1 + v_1)$.
3. Alice and Bob use Scalar Product Protocol on $(u_2, 1)$ and (r_2, r_2v_2) to get $z_2 = r_2(u_2 + v_2)$.
4. Alice computes $z = r\frac{z_1}{z_2} = \frac{u_1+v_1}{u_2+v_2}$, and sends it to Bob.

Protocol 6. *(Secure Two-Party Statistical Analysis Protocol in Homogeneous Model)*
Inputs: Alice has a data set $D_1 = ((x_1, y_1), \ldots, (x_k, y_k))$, Bob has another data set $D_2 = ((x_{k+1}, y_{k+1}), \ldots, (x_n, y_n))$,
Outputs: Alice and Bob both get $\bar{x}$, $\bar{y}$, r and b.

1. Alice sends $\sum_{i=1}^{k} x_i$ and $\sum_{i=1}^{k} y_i$ to Bob.
2. Bob sends $\sum_{i=k+1}^{n} x_i$ and $\sum_{i=k+1}^{n} y_i$ to Alice.
3. Alice and Bob both compute $\bar{x}$ and $\bar{y}$.
4. Alice computes $a_1 = \sum_{i=1}^{k} x_i \cdot y_i - k\bar{x}\bar{y}$, $a_2 = \sum_{i=1}^{k} x_i^2 - k\bar{x}^2$, and $a_3 = \sum_{i=1}^{k} y_i^2 - k\bar{y}^2$.
5. Bob computes $b_1 = \sum_{i=k+1}^{n} x_i \cdot y_i - (n-k)\bar{x}\bar{y}$, $b_2 = \sum_{i=k+1}^{n} x_i^2 - (n-k)\bar{x}^2$, and $b_3 = \sum_{i=k+1}^{n} y_i^2 - (n-k)\bar{y}^2$.
6. Using Scalar Product Protocol, Alice gets u_1 and u_2, while Bob gets v_1 and v_2, where $u_1 + v_1 = a_1^2 + 2a_1b_1 + b_1^2$ and $u_2 + v_2 = a_2a_3 + b_2a_3 + a_2b_3 + b_2b_3$.
7. Using Division Protocol, Alice and Bob gets $r^2 = \frac{u_1+v_1}{u_2+v_2}$ and $b = \frac{a_1+b_1}{a_2+b_2}$.

5 Summary and Future Work

In this paper, we have studied the problem of how to conduct the statistical analysis in a cooperative environment where neither of the cooperating parties wants to disclose its private data to the other party. Our preliminary work has shown that this problem, the secure two-party statistical analysis problem, could be solved in a way more efficient than the general circuit evaluation approach.

Apart from those basic statistical analysis computations studied in this paper, many other types of statistical analysis are also used in practice. A future direction would be to study more complicated statistical analysis computations, such as nonlinear regression, variance analysis and so on. Furthermore, we could also study, under the same secure two-party context, various data analysis computations other than the statistical analysis. Data mining is a very interesting and more complicated data analysis computation that is worth of study.

6 Acknowledgement

We thanks anonymous reviewers for their valuable comments.

References

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

[2] Wenliang Du, Mikhail J. Atallah and Florian Kerschbaum. Protocols for secure remote database access with approximate matching. Technical report, 2001.

[3] J. Benaloh. Dense probabilistic encryption. In *Proceedings of the Workshop on Selected Areas of Cryptography*, pages 120–128, Kingston, ON, May 1994.

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

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

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

[7] 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.

[8] 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.

[9] D. Naccache and J. Stern. A new cryptosystem based on higher residues. In *Proceedings of the 5th ACM Conference on Computer and Communications Security*, pages 59–66, 1998.

[10] 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.

[11] T. Okamoto and S. Uchiyama. An efficient public-key cryptosystem. In *Advances in Cryptology – EUROCRYPT 98*, pages 308–318, 1998.

[12] P. Paillier. Public-key cryptosystems based on composite degree residue classes. In *Advances in Cryptology – EUROCRYPT 99*, pages 223–238, 1999.

[13] A.C. Yao. How to generate and exchange secrets. In *Proceedings 27th IEEE Symposium on Foundations of Computer Science*, pages 162–167, 1986.