Privacy Preserving Distributed Permutation Test

Yunlong Mao
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing 210046, P. R. CHINA
njucsmyl@163.com

Yuan Zhang*
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing 210046, P. R. CHINA
zhangyuan05@gmail.com

ABSTRACT

In this paper, we propose a privacy-preserving algorithm for two-party distributed permutation test for the difference of means. Our algorithm allows two parties to jointly perform a permutation test on the union of their data without revealing their data to each other. Our algorithm is useful especially in areas where the testing data often contains private information e.g. clinic trial and biomedical research. We have proved the security of our algorithm and used experiment to show its efficiency. To the best of our knowledge, we are the first to address the privacy issues in permutation tests.

Keywords

Data confidentiality; Distributed computing; Statistical hypothesis test

1. INTRODUCTION

Compared with other statistical hypothesis tests such as t-test (Press, 1992), F-test (Box, 1953), etc., the permutation test (Fisher, 1935; Pitman, 1937) makes no assumption about the underlying population distribution and thus gives a more conservative result generally. Due to this reason, permutation tests become increasingly important in research areas which there exists a strong preference for controlling type I error such as medical trial and biomedical research.

A permutation test generally involves two or more groups of experimental units, which are exposed to different conditions. By applying the test, researchers are able to compare the outcomes of the groups and evaluate the effects of different conditions. Existing permutation test routines generally work for one party only. This party carries out experiments on each group, puts together the data and compares the outcomes using permutation tests. If more than one party is willing to apply permutation tests jointly on the union of their data, these parties either need to send their data to one single party and let that party perform the test, or rely on a regular distributed permutation test in which data is exchanged among all parties. However, both methods raise privacy concerns, especially in medical and biomedical research since data in these areas often includes private information of humans. Revealing private information to other parties may violate privacy regulations or laws like the Privacy Rule of the Health Insurance Privacy and Accountability Act (HIPAA). In order to address the privacy issues above, we need a permutation test algorithm that can be used by more than one party together and meanwhile protects the data privacy of each participating party.

In this paper, we study the problem of privacy preserving two-party distributed permutation test.(Extension to multiple parties is possible.) Consider, for example two fitness clinics who want to compare their treatments to see whether of them results in lower plasma cholesterol concentrations than the other. Both parties are interested in carrying out this study. However, neither clinic is willing to give its patients' data to the other because of privacy and security concerns. Our aim is to design an algorithm that can be performed jointly by both clinics to get the final testing result and meanwhile protects both clinics' data privacy throughout the entire process of the algorithm.

1.1 Paper Organization

The rest of this paper is organized as follows. We first give a brief description of the background knowledge in section 2. Then, we present our algorithm and analyze its security and efficiency in section 3. After discussing experimental evaluations in section 4, we conclude our paper in section 5.

2. BACKGROUND

Here we present the background knowledge of the concepts, and also techniques that are used in this paper.

2.1 Data confidentiality and privacy

The problem we are dealing with here is an example of the data confidentiality problem that lies in almost every area. In the area of statistics, or official statistics in particularly, federal statistical agencies such as Bureau of Labor Statistics (BLS), Census Bureau (CB), National Agricultural Statistics Service (NASS), etc. are required to protect

*Y.L. Mao, Y. Zhang(Corresponding Author) are with the State Key Laboratory for Novel Software Technology, Nanjing University, and also with the Computer Science and Technology Department, Nanjing University. This work was supported in part by NSFC-61321491, NSFC-61425024, NSFC-61300235, NSFC-61402223, and Jiangsu Province Double Innovation Talent Program.

SCC'16, May 30-June 03 2016, Xi'an, China

 ISBN 978-1-4503-4285-8/16/05…$15.00
DOI: http://dx.doi.org/10.1145/2898445.2898450

the confidentiality of their data subjects (Karr et al., 2005) when publishing their investigation results to the government or public. In the area of medical trial or biomedical research, the testing data often contains private information of individual person, therefore the confidentiality of data is strictly controlled under the Health Insurance Privacy and Accountability Act (HIPAA). Even in cases where no private information about the testing subjects is involved and no restriction comes from the outside world, the data owner herself may also wants to protect the data confidentiality since the data could contains the private information of the data owner e.g. business secrets, intellectual property, etc.

Although protecting data confidentiality can prevent loss caused by privacy disclosure, it makes utilizing or sharing the data more difficult at the same time. Such conflicts motivate the study of techniques that are able to balance the two goals including our work presented in this paper.

2.2 Problem formulation and definition of privacy

In this paper, we study the privacy issue involved in a distributed permutation test. In particular, we are dealing with scenarios in which the observed data of the two groups A and B are owned by two different parties P_0 and P_1 respectively. To get the significance test result P, both parties need to exchange their information. We aim to provide a secure algorithm that can be applied jointly by the two parties to get the final P-value and meanwhile their data is kept secure against each other.

We adopt the definition of privacy from the Cryptography literature, which is significantly stronger than the "privacy" considered in many other areas. For example, suppose in a two-party computation, party P^0 learns the sum of two numbers owned by another party P^1, but does not learn the two numbers themselves. In this case, it will be considered a violation of privacy by the definition we use here, although it may not be considered so by the standard of privacy in other areas.

Our solution is based on the *semi-honest model*, a standard security model widely used in privacy preserving data mining area (Agrawal and Srikant, 2000; Chen and Liu, 2005; Chen and Zhong, 2009; Heer, 1993; Laur et al., 2006; Lindell and Pinkas, 2000; Vaidya et al., 2008) and also in the existing work that study privacy preserving statistical tests (Chen and Zhong, 2011; Du and Atallah, 2001; Du et al., 2004). This model assumes that participating parties are "honest-but-curious". That is, each party follows the protocol with no deviation, but can attempt to derive other parties' private information.

Accordingly, privacy means that each participating party learns only the final result of the test, and nothing more about other parties' data after she performs the algorithm. Below we adopt the standard simulation paradigm (Goldreich, 2004) in Cryptography to define the privacy formally. The basic logic of this paradigm is that, if whatever a party observes during the execution of an algorithm could be essentially computed from that party's private input and output (and also its coin flips if the algorithm is probabilistic), then the algorithm can be considered as privacy-preserving.

Recall that P_0 has private input A and P_1 has private input B. We assume the total number of testing subjects is known to both parties and belongs in the two parties' public input t. Let p be the final P-value of the permutation test. Denote by $VIEW_1(A, B, t)$ (resp., $VIEW_2(A, B, t)$) the view of A (resp., B) which includes A's (resp., B's) private input, random coin flips, and received messages while participating the algorithm. We use $\stackrel{c}{\equiv}$ to denote *computational indistinguishability* (Goldreich, 2001).

Definition 1. A two-party distributed permutation test algorithm that outputs the final P-value is *privacy preserving* if there exist probabilistic polynomial-time simulators $\mathcal{PPM}_1$ and $\mathcal{PPM}_2$ that simulate the view of P_1 and P_2 respectively, such that

$$\{\mathcal{PPM}_1(A, t, p)\}_{A,t} \stackrel{c}{\equiv} \{VIEW_1(A, B, t)\}_{A,t},$$

$$\{\mathcal{PPM}_2(B, t, p)\}_{B,t} \stackrel{c}{\equiv} \{VIEW_2(A, B, t)\}_{B,t}.$$

2.3 Privacy preserving data mining

One line of research closely related to our problem is privacy preserving data mining. Research in this area mainly focus on protecting data's privacy in the mining process. Solutions can be divided into two categories in general. Solutions in the first category use data perturbation techniques such as adding noises to the data (Agrawal and Srikant, 2000), performing rotations on the data (Chen and Liu, 2005) and replacing the original data by samples from the same distribution (Heer, 1993), to protect data privacy. Solutions in the second category use cryptographic techniques to protect the data and perform the computation. In both categories, many privacy preserving data mining algorithms (Agrawal and Srikant, 2000; Chen and Liu, 2005; Chen and Zhong, 2009; Heer, 1993; Laur et al., 2006; Lindell and Pinkas, 2000; Vaidya et al., 2008) have been proposed. One major difference between these two kinds of solutions is their computational costs and accuracy. The first type of solutions generally have a lower computational cost but lose some accuracy. On the contrary, the second ones usually achieve 100% accuracy but require higher computational cost. Our solution in this paper is similar to solutions in the second category in that it uses cryptographic techniques. However, the problem we study is not a data mining problem.

Till now, there are only a handful of papers that study the privacy issues in statistical tests. In (Du and Atallah, 2001) and (Du et al., 2004), two-party privacy preserving algorithms are proposed for univariate and multivariate linear regression problems respectively. In (Chen and Zhong, 2011), a privacy preserving algorithm is proposed to compare survival curves using logrank test. However, none of existing works has considered permutation test. To the best of our knowledge, we are the first to give secure solutions to the problem of privacy preserving permutation test.

2.4 Homomorphic encryption and two-party threshold decryption

To protect the data and compute the final result securely, we use a public-key cryptosystem with homomorphic encryption in our algorithm. The cryptosystem is a variant of Elgamal cryptosystem (ElGamal, 1985). It is semantically secure under the standard Decisional Diffie-Hellman (DDH) Assumption (Boneh, 1998). Denote this cryptosystem by $\mathcal{E}$, and the encryption of an arbitrary plaintext m by $\overline{m}$. Below is a brief description of $\mathcal{E}$:

- The plaintext space $\mathcal{M}$ is a large field Z_q of size $\Theta(2^k)$ (q is a k-bit large prime number).

- Given a random number k_{pri} and a generator g of Z_q as the private key (k_{pri}, g), the corresponding public key can be calculated as

$$(k_{pub}, g) = (g^{k_{pri}}, g).$$

- Given a public key k_{pub}, a random number r, $\mathcal{E}$ encrypts a plaintext $m \in \mathcal{M}$ as

$$\overline{m} = E(m, r) \stackrel{c}{=} (M, R) = \big(g^m (k_{pub})^r, g^r\big).$$

- $\mathcal{E}$ is *additively homomorphic*. $\mathcal{E}$ provides an efficient "ciphertext-addition" operation $+_h$ which can calculate the ciphertext of $m_1 + m_2$ for any $m_1, m_2 \in \mathcal{M}$ using the ciphertexts of these two plaintexts without decryption:

$$\begin{aligned}\overline{m_1 + m_2} =& \overline{m_1} +_h \overline{m_2} \\ =& (M_1, R_1) +_h (M_2, R_2) \\ =& (M_1 M_2, R_1 R_2).\end{aligned}$$

- $\mathcal{E}$ provides an efficient "ciphertext-multiplication" operation $\times_h$ which can compute the ciphertext of the product $a \times m_1$ for any $m_1 \in \mathcal{M}$ and constant a as

$$\overline{a \times m_1} = a \times_h \overline{m_1} = (M_1^a, R_1^a).$$

- Given a ciphertext $\overline{m} = (M, R)$ and the private key k_{pri}, $\mathcal{E}$ decrypts g^m as

$$g^m = M / R^{k_{pri}}.$$

Furthermore, we want the cryptosystem to be a two-party threshold public-key system. In such a system, given a common public key for encryption, the corresponding private key is divided into two parts and distributed between two parties P_1 and P_2. Neither party is able to decrypt a ciphertext without the help of the other one. To achieve this, P_1 and P_2 first agree on the values of q and g, and then choose k_{pri1} and k_{pri2} randomly and independently. After calculating the corresponding k_{pub1} and k_{pub2} and exchanging them, both parties use $(g, k_{pub1} k_{pub2})$ as the common public key. To decrypt a ciphertext $\overline{m} = (M, R)$, P_1 and P_2 could conduct a "threshold-decryption" operation as follows. First, P_2 sends $R^{k_{priv2}}$ to P_1. Then P_1 computes g^m as $M/(R^{k_{pri1}} R^{k_{pri2}})$ and sends it back to P_2. After that both parties can find out the plaintext m by themselves.

2.5 Secure comparison algorithm

In our solution, we apply a secure comparison algorithm to compare two numbers securely. Suppose there are two parties P_1 and P_2. P_1 has a private number s and P_2 has a private number t. Both parties would like to compute whether $s > t$. Let $\mathcal{C}(s, t)$ represent a comparison between s and t, and r be comparison result such that

$$\mathcal{C}(s, t) = r = \begin{cases} 1 & \text{if } s > t \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

To perform comparisons securely, we use a secure comparison algorithm $\mathcal{SC}(s, t)$ that satisfies the following conditions:

- After both parties use their private number as the input, the algorithm returns an encryption of 1 if s is greater than t, or an encryption of zero otherwise.
- Besides the encryption of the comparison result, each party knows no more knowledge about other party's input.

In (Damgard et al., 2008), Damgard et al. propose a secure comparison algorithm in the semi-honest model based on the same homomorphic cryptosystem as we introduced in the previous section. Let the binary presentations of s and t be $s_l, \cdots, s_1$ and $t_l, \cdots, t_1$ where s_1 and t_1 are the least significant bits. Their algorithm is based on the following fact: if s is greater than t, then there is a "pivot bit" i such that $t_i - s_i + 1 = 0$ and $s_j \oplus t_j = s_j + t_j - 2 s_j t_j$ is zero for every $i < j \leq l$, and vice versa. Their algorithm uses homomorphic properties to do all arithmetic computations using encryptions of the bits and verifies if such pivot bit exists. It has been proved that this secure comparison algorithm is secure in the semi-honest model.

Besides providing the encryption of the final comparison result, Damgard et al. also show how use Toft's method (Toft, 2007) to adapt their secure comparison algorithm to provide the final result as the exclusive-OR of two secret shares (Damgard et al., 2008). For ease of presentation, we use $\mathcal{SC}(s, t)$ to represent the basic secure comparison algorithm returning the result in its encrypted ciphertext $\overline{r}$, and use $\mathcal{SC}'(s, t)$ to represent the variant outputting (r_1, r_2) such that the comparison result is the exclusive-OR of two shares r_1 (only known to P_1) and r_2 (only known to P_2).

3. PRIVACY PRESERVING ALGORITHM FOR DISTRIBUTED PERMUTATION TEST

In this section, we present our privacy preserving algorithm for two-party distributed permutation test on the difference of means.

Suppose n subjects are assigned to two treatments or experiments, where n_1 subjects to treatment T_a and the rest n_2 subjects to treatment T_b (Here $n = n_1 + n_2$.). A common numerical result that evaluates the effect of a treatment is collected from every subject. Denote by $A = \{a_1, a_2, \ldots, a_{n_1}\}$ (resp. $B = \{b_1, b_2, \ldots, b_{n_2}\}$) as the collection of the results of T_a (resp. T_b). The null hypothesis (H_0) is that different treatments make no significant difference or the observed difference of means on different groups is merely a matter of chance.

Given any set C, let $\Sigma(C)$ denote the sum of all elements in C. The observed difference of means between A and B is calculated as

$$D_m = \Sigma(A)/n_1 - \Sigma(B)/n_2. \tag{2}$$

Let X_A be a subset of A and X_B be a subset of B such that X_A and X_B have the same cardinality. A "valid" permutation is generated by switching all elements between X_A and X_B. The difference of means under this permutation is calculated as

$$\begin{aligned} d_m(X_A, X_B) =& \big(\Sigma(A) - \Sigma(X_A) + \Sigma(X_B)\big)/n_1 \\ & - \big(\Sigma(B) - \Sigma(X_B) + \Sigma(X_A)\big)/n_2. \end{aligned} \tag{3}$$

Given a combination of (X_A, X_B), its corresponding permutation is considered to have a same or more extreme outcome if

$$|d_m(X_A, X_B)| \geq |D_m|. \tag{4}$$

Define a function

$$\delta(X_A, X_B) = \begin{cases} 1 & \text{if } |d_m(X_A, X_B)| \geq |D_m| \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

Then the two-sided P-value can be calculated by examining all possible $(X_A,\ X_B)$ combinations:

$$P = \sum_{(X_A, X_B)} \delta(X_A, X_B) / \big(n!/(n_1)!(n_2)!\big). \quad (6)$$

When treatment T_a and treatment T_b are conducted by two independent parties P_1 and P_2 respectively, P_1 owns T_a's result $A = \{a_1, a_2, \ldots, a_{n_1}\}$ and P_2 owns T_b's result $B = \{b_1, b_2, \ldots, b_{n_2}\}$. In the remainder of this section, we show how the two parties can apply our algorithm to perform the permutation test above without revealing the data that they own to each other.

The main procedure of our algorithm is as follows.

- P_1 and P_2 jointly generate all possible subset pairs (X_A, X_B) such that X_A and X_B have the same cardinality first.
- For each pair, P_1 and P_2 use secure comparison algorithms to verify whether the corresponding permutation has a same or more extreme outcome. The verification result $\delta(X_A, X_B)$ is encrypted and both parties know the ciphertext $\overline{\delta(X_A, X_B)}$ only.
- After knowing every $\overline{\delta(X_A, X_B)}$, P_1 and P_2 can compute the encryption of the total number of permutations that have same or more extreme outcomes. This is done by using the additively homomorphic property to compute the encryption of all $\delta(X_A, X_B)$'s sum from the encryptions of all $\delta(X_A, X_B)$.
- Finally, P_1 and P_2 jointly decrypt the sum and compute the corresponding P-value.

Below we give detailed explanations of how each step can be carried out.

3.1 How to generate all possible (X_A, X_B) pairs securely

Recall that we have assumed both parties know the total number of data $n(n = n_1+n_2)$, this means each party knows the number of data the other party has. Let n_{min} be the smaller one between n_1 and n_2.

To generate all possible (X_A, X_B) pairs, each party first lists her all subsets of cardinality i for $i = 0, 1, \ldots, n_{min}$.

Then for each i, P_1 traverses all subset X_A of cardinality i, and writes each subset for $n_2!/i!(n_2-i)!$ times on her "X_A-list" L_1. For each i, P_2 traverses all subset X_B of cardinality i for $n_1!/i!(n_1-i)!$ iterations. Every time P_2 reads a subset, P_2 writes it on her "X_B-list" L_2.

After this procedure, it is easy to verify that both parties have a list that contains $N!/(n_1)!(n_2)!$ subsets. And by traversing their lists simultaneously, every time P_0 visits a subset X_A, P_1 would be visiting an X_B of the same cardinality. All possible (X_A, X_B) pairs can be traversed in the end.

Since both parties generate their lists by themselves, this procedure is straightforwardly secure.

3.2 How to securely compute $\overline{\delta(X_A, X_B)}$

For each (X_A, X_B), P_1 and P_2 need to compute the encryption of $\delta(X_A, X_B)$. To do this, P_1 and P_2 need to securely verify whether $|d_m(X_A, X_B)|$ is no less than $|D_m|$ which is equivalent to verify whether

$$(n_1 n_2)^2 (d_m(X_A, X_B) + D_m)(d_m(X_A, X_B) - D_m) \geq 0. \quad (7)$$

Use $\mathcal{C}(\cdot,\cdot)$ to represent a comparison algorithm as (1) and four bit-numbers r_1, r_2, r_3 and r_4 to represent the results of the following four comparisons :

$$\begin{array}{l} r_1 = \mathcal{C}\big(2n_2\Sigma(A) - n\Sigma(X_A), 2n_1\Sigma(B) - n\Sigma(X_B)\big), \\ r_2 = \mathcal{C}\big(\Sigma(A), \Sigma(B)\big), \\ r_3 = \mathcal{C}\big(2n_1\Sigma(B) - n\Sigma(X_B), 2n_2\Sigma(A) - n\Sigma(X_A)\big), \\ r_4 = \mathcal{C}\big(\Sigma(B), \Sigma(A)\big). \end{array} \quad (8)$$

Below we prove that predicate (7)'s result (i.e. $\delta(X_A, X_B)$) can be computed from these four bit-numbers.

THEOREM 1. $\delta(X_A, X_B) = 1 - r_1 \times r_2 - r_3 \times r_4$.

Proof: Put Dm's definition (2) and d_m's definition (3) into the inequality (7), the predicate is equivalent to

$$\big(2n_2\Sigma(A) - n\Sigma(X_A) - 2n_1\Sigma(B) + n\Sigma(X_B)\big) \times \big(\Sigma(X_B) - \Sigma(X_A)\big) \geq 0 \quad (9)$$

Instead of verifying whether (9) is true, it is equivalent to check whether its negation is true. It is easy to see that the negation

$$\big(2n_2\Sigma(A) - n\Sigma(X_A) - 2n_1\Sigma(B) + n\Sigma(X_B)\big) \times \big(\Sigma(X_B) - \Sigma(X_A)\big) < 0 \quad (10)$$

is equivalent to

$$\begin{cases} 2n_2\Sigma(A) - n\Sigma(X_A) > 2n_1\Sigma(B) - n\Sigma(X_B) \\ \Sigma(A) > \Sigma(B) \end{cases} \quad (11)$$

OR

$$\begin{cases} 2n_1\Sigma(B) - n\Sigma(X_B) > 2n_2\Sigma(A) - n\Sigma(X_A) \\ \Sigma(B) > \Sigma(A) \end{cases}. \quad (12)$$

Use 1 to represent true and 0 to represent false, predicate (11)'s result equals to $r_1 \times r_2$ and predicate (12)'s result equals to $r_3 \times r_4$. It is easy to see predicates (11) and (12) cannot be both true. Therefore, the result of (11) OR (12) can be written as the sum of $r_1 \times r_2$ and $r_3 \times r_4$.

Since $\delta(X_A, X_B)$ equals to the result of predicate (7) which is the negation of predicate (10), it is straightforward to see

$$\delta(X_A, X_B) = 1 - r_1 \times r_2 - r_3 \times r_4. \quad (13)$$

□

Given (13), if the ciphertexts of $r_1 \times r_2$ and $r_3 \times r_4$ can be securely computed, the ciphertext of $\delta(X_A, X_B)$ can be easily computed using the homomorphic property of the cryptosystem:

$$\overline{\delta(X_A, X_B)} = \overline{1} +_h (-1) \times_h \overline{r_1 \times r_2} +_h (-1) \times_h \overline{r_3 \times r_4}. \quad (14)$$

Below we show how P_1 and P_2 securely compute the $\overline{r_1 \times r_2}$. $\overline{r_3 \times r_4}$ can be computed in a similar way.

First, P_1 computes $2n_2\Sigma(A) - n\Sigma(X_A)$ and $\Sigma(A)$ based on A and X_A (A and X_A are only known by P_1). Similarly, P_2 computes $2n_1\Sigma(B) - n\Sigma(X_B)$ and $\Sigma(B)$ based on B and X_B (B and X_B are only known by P_2).

Second, P_1 and P_2 jointly perform the secure comparison algorithm $\mathcal{SC}(\cdot,\cdot)$ to compare two private numbers $2n_2\Sigma(A) -$

$n\Sigma(X_A)$ and $2n_1\Sigma(B) - n\Sigma(X_B)$. $\mathcal{SC}(\cdot,\cdot)$ outputs the encryption of the comparison result r_1 to both parties:

$$\overline{r_1} = \mathcal{SC}\big(2n_2\Sigma(A) - n\Sigma(X_A), 2n_1\Sigma(B) - n\Sigma(X_B)\big). \quad (15)$$

Third, P_1 and P_2 jointly perform the secure comparison algorithm $\mathcal{SC}'(\cdot,\cdot)$ to compare two private numbers $\Sigma(A)$ and $\Sigma(B)$. Different from $\mathcal{SC}$, $\mathcal{SC}'$ outputs two random bit-shares r_{2a} and r_{2b} of the comparison result r_2 such that r_{2a} (resp. r_{2b}) is only to P_1 (resp. P_2) and r_2 equals to the exclusive-OR of r_{2a} and r_{2b}:

$$(r_{2a}, r_{2b}) = \mathcal{SC}'\big(\Sigma(A), \Sigma(B)\big). \quad (16)$$

Fourth, P_1 and P_2 jointly compute the encryption of $r_1 \times r_2$ based on the encryption of r_1 and random bit-shares of r_2 using the homomorphic property of the cryptosystem. Since

$$\begin{aligned} r_2 &= r_{2a} \otimes r_{2b} \\ &= r_{2a} + r_{2b} - 2r_{2a}r_{2b}, \end{aligned} \quad (17)$$

the ciphertext of $r_1 \times r_2$ can be computed as

$$\begin{aligned} \overline{r_1 \times r_2} &= \overline{r_{2a} \times r_1 + r_{2b} \times r_1 + (-2r_{2a}r_{2b}) \times r_1} \\ &= \overline{r_{2a} \times r_1} +_h \overline{r_{2b} \times r_1} +_h \overline{(-2r_{2a}r_{2b}) \times r_1} \\ &= r_{2a} \times_h \overline{r_1} +_h r_{2b} \times_h \overline{r_1} \\ &\quad +_h (-2r_{2a}) \times_h (r_{2b} \times_h \overline{r_1}). \end{aligned} \quad (18)$$

In particular, $r_{2a} \times_h \overline{r_1}$ is computed by P_1 locally based on her knowledge of r_{2a} and $\overline{r_1}$. Similarly $r_{2b} \times_h \overline{r_1}$ is computed by P_2 locally. P_2 sends $r_{2b} \times_h \overline{r_1}$ to P_1, then P_1 computes $(-2r_{2a}) \times_h (r_{2b} \times_h \overline{r_1})$, and $\overline{r_1 \times r_2}$. After P_1 sends $\overline{r_1 \times r_2}$ back, both parties know the ciphertext of $r_1 \times r_2$.

After $\overline{r_1 \times r_2}$ and $\overline{r_3 \times r_4}$ are known to both parties, they can compute the encryption of $\delta(X_A, X_B)$ locally as (14).

The security of the secure comparison algorithms guarantees that no private information is disclosed during the comparisons. After the secure comparisons, each party receives only ciphertexts of unknown plaintexts from the other party. The semantic security of the cryptosystem guarantees no private information is disclosed.

3.3 How to securely compute the P-value

While P_1 and P_2 traverse all possible (X_A, X_B) pairs, they jointly compute the corresponding $\overline{\delta(X_A, X_B)}$. Using the homomorphic property, both parties can compute the ciphertext of $\Sigma_{(X_A,X_B)}\delta(X_A, X_B)$. After $\overline{\Sigma_{(X_A,X_B)}\delta(X_A, X_B)}$ is computed, P_1 and P_2 jointly apply the threshold decryption algorithm to decrypt the ciphertext and get the value of $\Sigma_{(X_A,X_B)}\delta(X_A, X_B)$. Using (6), the final P-value can be computed locally.

The security of this part is guaranteed by the security of the threshold decryption algorithm.

3.4 Algorithm summary

We summarize the main steps of our privacy-preserving permutation test algorithm as follows.

PPPT(P_1, P_2, A, B)
P_1's **private input**: $A = \{a_1, a_2, \ldots, a_{n_1}\}$.
P_2's **private input**: $B = \{b_1, b_2, \ldots, b_{n_2}\}$.
pubic input: n_1, n_2, $n = n_1 + n_2$ and $n_{min} = min\{n_1, n_2\}$.
init: both parties compute $\overline{\Sigma} = E(0, 1)$.
for $i = 0 : n_{min}$ **do:**
  P_1 enumerates A's every subset X_A of cardinality i.
  for every X_A **do:**
    P_2 enumerates B's every subset X_B of cardinality i.
    for every X_B **do:**
      P_1 and P_2 jointly compute $\overline{\delta(X_A, X_B)}$.
      both parties compute $\overline{\Sigma} \leftarrow \overline{\Sigma} +_h \overline{\delta(X_A, X_B)}$.
    end for
  end for
end for
P_1 and P_2 jointly decrypt the ciphertext $\overline{\Sigma}$ and get Σ.
output: both parties compute the two-sided P-value as $P = \Sigma/(n!/(n_1)!(n_2)!)$

3.5 Security analysis

In this section, we formally prove our algorithm is privacy-preserving assuming the total number of test subjects is publicly known.

THEOREM 2. The distributed algorithm Privacy Preserving Permutation Test (hereinafter referred to as **PPPT**) is privacy preserving assuming that the total number of test subjects is publicly known.

Proof: According to the Composition Theorem for the semi-honest model (Goldreich, 2004), it is enough to prove our algorithm is secure after replacing each secure comparison algorithm call and the threshold decryption algorithm call in our algorithm with a corresponding "oracle call" that takes P_1's and P_2's inputs and returns the results. We construct simulators to simulate the views of the parties in the oracle-aided algorithm.

We construct $\mathcal{PPM}_1$ as follows. Given the final output p, the publicly known input n and the private A, $\mathcal{PPM}_1$ first generates the same subset list L_1 locally as **PPPT** does. For each subset X_A in the list, $\mathcal{PPM}_1$ first generates a random encryption of a random plaintext to simulate the the $\mathcal{SC}$-oracle's output $\overline{r_1}$, uses a random bit to simulate the $\mathcal{SC}'$-oracle's output r_{2a} and uses a random encryption of a random plaintext to simulate the received $r_{2b} \times_h \overline{r_1}$. Then $\mathcal{PPM}_1$ similarly uses two random encryptions of two random plaintexts to simulate $\overline{r_3}$ and $r_{4b} \times_h \overline{r_3}$, and uses a random bit number to simulate r_{4a}. Finally $\mathcal{PPM}_1$ uses p to simulate the threshold-decryption-oracle's output. The computational indistinguishability is guaranteed by the semantic security of the cryptosystem, the security of the secure comparison algorithms and the security of the threshold decryption algorithm.

Similarly, we can construct $\mathcal{PPM}_2$ as follows. Given the final output p, the publicly known input n and the private B, $\mathcal{PPM}_2$ first generates the same subset list L_2 locally as **PPPT** does. For each subset X_B in the list, $\mathcal{PPM}_2$ first generates a random encryption of a random plaintext to simulate the the $\mathcal{SC}$-oracle's output $\overline{r_1}$, uses a random bit to simulate the $\mathcal{SC}'$-oracle's output r_{2b} and uses a random encryption of a random plaintext to simulate the received $\overline{r_1 \times r_2}$. Then $\mathcal{PPM}_2$ similarly uses two random encryptions of two random plaintexts to simulate $\overline{r_3}$ and $\overline{r_3 \times r_4}$, and uses a random bit number to simulate r_{4b}. Finally $\mathcal{PPM}_2$ uses p to simulate the threshold-decryption-oracle's output. The computational indistinguishability is guaranteed by the semantic security of the cryptosystem, the security of the secure comparison algorithms and the security of the threshold decryption algorithm. □

3.6 Efficiency analysis

Our algorithm's efficiency can be estimated in terms of the secure comparison algorithm it uses since the secure comparison algorithm is the most computation-intensive operation in our algorithm.

Given sample sizes n_1 and n_2, our algorithm needs to compute $(n)!/n_1!n_2!$ ciphertexts of $\delta(X_A, X_B)$. In total, this requires $2(n)!/n_1!n_2!$ secure comparisons on the corresponding $(2n_2\Sigma(A) - n\Sigma(X_A), 2n_1\Sigma(B) - n\Sigma(X_B))$-pairs and $(2n_1\Sigma(B) - n\Sigma(X_B), 2n_2\Sigma(A) - n\Sigma(X_A))$-pairs, as well as two secure comparisons on $(\Sigma(A), \Sigma(B))$-pair and $(\Sigma(B), \Sigma(A))$-pair. Since the efficiency difference of (SC) and (SC') is very small, we do not differentiate them here.

Therefore, our algorithm's efficiency can be estimated as $2(n)!/n_1!n_2!+2$ times of secure comparison algorithms. Note that the exponential complexity in n is introduced by the permutation test algorithm itself, not caused by our secure algorithm.

4. EXPERIMENTAL EVALUATION

To give an intuitive idea of the performance of our algorithm, we perform it using an example from (Ludbrook and Dudley, 1998). The data set is from a hypothetical experiment that compares the differential effects of two treatments on plasma cholesterol level in patients. Here we assume treatment 1 (eating fish but not meat) and treatment 2 (eating meat but not fish) are conducted by two separate fit clinics on their patients. Accordingly, patients' test data are owned separately by the two clinics. Below we list the data (patients' plasma cholesterol concentrations).

Treatment 1(Fish): 5.42, 5.86, 6.16, 6.55, 6.80, 7.00, 7.11
Treatment 2(Meat): 6.51, 7.56, 7.61, 7.84, 11.50

The algorithm is implemented using C++ and complied with gcc version 4.5.2. Experiments are conducted on a laptop running Ubuntu 11.04 with dual 2.33 GHz Intel T7600 CPUs and 2 GB memory. All cryptographic operations are implemented using Crypto++ Library 5.6.1 (Dai, 2010). We summarize our experimental results as follows.

(n_1, n_2):	(7, 5)
total permutations:	792
P-value by (Ludbrook and Dudley, 1998):	7/792
P-value by our algorithm:	7/792
k_p's length (bit):	512
effective bits l:	32
total computational cost (sec):	1108.5

5. CONCLUSION

In this paper, we propose a privacy preserving algorithm for two-party distributed permutation test. We prove the security of our algorithm and use experiment to evaluate the correctness and efficiency of our algorithm. Assuming the total number of test subjects is publicly known, our algorithm can correctly compute the P-value without revealing any additional information to each participating party.

References

Agrawal and Srikant (2000). Privacy-preserving data mining. *SIGMODREC: ACM SIGMOD Record*, 29.

Boneh, D. (1998). The decision diffie-hellman problem. *Algorithmic Number Theory*, pages 48–63.

Box, G. (1953). Non-normality and tests on variances. *Biometrika*, 40(3/4):318–335.

Chen, K. and Liu, L. (2005). Privacy preserving data classification with rotation perturbation. In *Proceedings of the Fifth IEEE International Conference on Data Mining*, pages 589–592. IEEE Computer Society.

Chen, T. and Zhong, S. (2009). Privacy-preserving backpropagation neural network learning. *IEEE Transactions on Neural Networks*, 20(10):1554–1564.

Chen, T. and Zhong, S. (2011). Privacy-preserving models for comparing survival curves using the logrank test. *Computer methods and programs in biomedicine*.

Dai, W. (2010). Crypto++ library. *URL: http://www. cryptopp. com*.

Damgard, I., Geisler, M., and Kroigard, M. (2008). Homomorphic encryption and secure comparison. *International Journal of Applied Cryptography*, 1(1):22–31.

Du, W. and Atallah, M. (2001). Privacy-preserving cooperative statistical analysis. In *acsac*, page 0102. Published by the IEEE Computer Society.

Du, W., Han, Y., and Chen, S. (2004). Privacy-preserving multivariate statistical analysis: Linear regression and classification. In *Proceedings of the 4th SIAM International Conference on Data Mining*, volume 233. Lake Buena Vista, Florida.

ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology*, pages 10–18. Springer.

Fisher, R. (1935). The design of experiments.

Goldreich, O. (2001). *Foundations of cryptography: Basic Tools*, volume 1. Cambridge Univ Pr.

Goldreich, O. (2004). *Foundations of cryptography: Basic applications*, volume 2. Cambridge Univ Pr.

Heer, G. (1993). A bootstrap procedure to preserve statistical confidentiality in contingency tables. In *Proceedings of the International Seminar on Statistical Confidentiality*, pages 261–271.

Karr, A., Lin, X., Sanil, A., and Reiter, J. (2005). Secure regression on distributed databases. *Journal of Computational and Graphical Statistics*, 14(2):263–279.

Laur, S., Lipmaa, H., and Mielikäinen, T. (2006). Cryptographically private support vector machines. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 618–624. ACM.

Lindell, Y. and Pinkas, B. (2000). Privacy preserving data mining. In Bellare, M., editor, *CRYPTO*, volume 1880 of *Lecture Notes in Computer Science*, pages 36–54. Springer.

Ludbrook, J. and Dudley, H. (1998). Why permutation tests are superior to t and f tests in biomedical research. *American Statistician*, pages 127–132.

Pitman, E. (1937). Significance tests which may be applied to samples from any populations. *Supplement to the Journal of the Royal Statistical Society*, 4(1):119–130.

Press, W. (1992). *Numerical recipes in FORTRAN: the art of scientific computing*, volume 1. Cambridge Univ Pr.

Toft, T. (2007). *Primitives and applications for multi-party computation*. PhD thesis, University of Aarhus, Aarhus, Denmark.

Vaidya, J., Kantarcioglu, M., and Clifton, C. (2008). Privacy-preserving naïve bayes classification. *VLDB J.*, 17(4):879–898.