Hindawi
Security and Communication Networks
Volume 2017, Article ID 5207386, 11 pages
https://doi.org/10.1155/2017/5207386

Research Article

Efficient Solutions to Two-Party and Multiparty Millionaires' Problem

Xin Liu,[1,2] Shundong Li,[1] XiuBo Chen,[3] Gang Xu,[4] Xiaolin Zhang,[2] and Yong Zhou[5]

[1]*School of Computer Science, Shaanxi Normal University, Xi'an 710062, China*
[2]*School of Information Engineering, Inner Mongolia University of Science and Technology, Baotou 014010, China*
[3]*Information Security Center, State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China*
[4]*School of Software Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China*
[5]*Information and Network Center, Northwest University of Politics and Law, Xi'an 710061, China*

Correspondence should be addressed to Shundong Li; shundong@snnu.edu.cn

Received 13 December 2016; Revised 30 March 2017; Accepted 18 April 2017; Published 25 May 2017

Academic Editor: Mamoun Alazab

The millionaires' problem is the basis of secure multiparty computation and has many applications. Using a vectorization method and the Paillier encryption scheme, we first propose a secure two-party solution to the millionaires' problem, which can determine $x = y$, $x < y$, or $x > y$ in one execution. Subsequently, using the vectorization and secret splitting methods, we propose an information-theoretically secure protocol to solve the multiparty millionaires' problem (a.k.a. secure sorting problem), and this protocol can resist collusion attacks. We analyze the accuracy and security of our protocols in the semihonest model and compare the computational and communication complexities between the proposed protocols and the existing ones.

1. Introduction

The millionaires' problem (abstracted as a greater than (GT) problem) was proposed by Yao [1] as follows: two millionaires, Alice and Bob, want to know who has more wealth, but they do not want to disclose their wealth value to each other. The GT problem can be applied to a practical situation: suppose that Alice wants to buy some commodities from Bob online, and she hopes to pay \$$x$ at most, while Bob wants to sell the commodities for \$$y$, but neither wants to disclose this price. Therefore, they need to compare x and y privately. If $x > y$, they will make a bargain; otherwise, they do not waste their time trying to reach an agreement upon price. The GT problem has also been applied to secure multiparty computation (SMC), a focus in the international cryptography community [2]. Goldwasser [3] predicted that SMC will become an integral part of our computing reality in the future because of its rich theory and potential as a powerful tool.

Motivated by Goldwasser's predictions, cryptographic researchers have studied many SMC problems, including private sorting problems [4], private determination of the relationships among sets [5] and geometry [6], private voting problems [7], and private data mining [8]. Goldreich [9] studied theoretical SMC problems and established a simulation paradigm that has been extensively used to evaluate the security of SMC protocols.

The GT problem is a foundational component of many SMC protocols [10–15] and should be solved efficiently for many applications. The secure two-party computation of the GT problem can be applied to securely solve the rational interval computation [16], which can further determine the inclusion problems between point and ring, point and infinite region, point and segment, and so on and even can be used to reduce the cost in real commodity transaction. Protocols for the GT problem can be used to determine graphics similarity [17]. SMC protocols of the GT problem were used to solve secure computation of skyline query in mapreduce [18], secure stable matching at scale [19], private large-scale databases [20], association rule mining [21], and so forth. In addition, Bogdanov et al. [22] proposed a tool for cryptographically secure statistical analysis based on a sorting method. Aly and Van Vyve [23] proposed a secure

sorting protocol for secure single-commodity multimarket auctions. Blanton and Aguiar [24] applied a sorting protocol to compute multiset operations.

Cryptographic researchers have proposed several GT protocols. Yao [1] used garbled circuits for solving the GT problem. Grigoriev and Shpilrain [30] used various laws of classical physics to offer several solutions of Yao's millionaires' problem. Chang et al. [31] proposed a pioneering quantum private comparison protocol for some users.

Many researchers have proposed GT protocols based on homomorphic encryption schemes. Li and Wang [32] used the Paillier encryption scheme to compare two encoding numbers. Blake and Kolesnikov [25] used the additive homomorphism of the Paillier encryption scheme to construct a two-round GT protocol with the computational cost of $((4b + 1)\lg N + 6b)$ modular multiplications, where b is the bit number of the private inputs and N is the modulus of the Paillier encryption scheme. Lin and Tzeng [26] proposed a two-round protocol using the multiplicative homomorphism of the ElGamal encryption scheme, with the computational cost of $(5b \lg N + 4b - 6)$ modular multiplications. Cao and Liu [33] improved the Lin-Tzeng solution to Yao's millionaires problem by having the two participators alternately perform the original Lin-Tzeng protocol twice.

In addition, some researchers proposed SMC protocols for the GT problem based on the linear secret sharing method. Nishide and Ohta [29] constructed a simplified bit-decomposition protocol for the GT problem, whose communication cost is 15 rounds. The protocol cannot distinguish $x = y$ from $x < y$. Damgard et al. [27] proposed a novel technique to convert a polynomial sharing of the secret into a sharing of the bits of inconstant rounds. Its computation complexity is $O(l \log l)$ (l is the bit number of the shared value) modular multiplications.

Although several protocols have been proposed, their efficiency is not very high and more efficient SMC protocols are desirable. Therefore, this paper proposes two efficient solutions to both the two-party and multiparty GT problems.

Our Contributions. (1) Using the vectorization method, we encode a number into a vector and transform the GT problem into a vector-element-selection problem.

(2) Protocol 1 securely solves the two-party GT problem based on the vectorization method and the Paillier encryption scheme. It determines $x = y$, $x < y$, or $x > y$ in one execution. The computational cost is $(2(s + 2)\lg N)$ modular multiplications (N is the modulus of the Paillier encryption scheme; s is the encoding vector dimension), and the communication cost is one round. We compare the computational and communication complexities between Protocol 1 and the existing solutions in Section 4.

(3) Protocol 2 securely computes the multiparty GT problem (i.e., secure sorting problem) based on the vectorization method and the secret splitting method without any encryption scheme based on computational assumptions. The computational cost is negligible and the communication cost is $2n(n - 1)$ rounds (n is the number of parties). It can resist collusion attacks and is information-theoretically secure. We compare the computational and communication complexities between Protocol 2 and the existing solutions in Section 4.

Organization. Section 2 introduces related definitions and methods, including the SMC model, the semihonest party, the simulation paradigm, the Paillier encryption scheme, and the vectorization method. Section 3 proposes the two-party and multiparty GT protocols, analyzes the protocols' accuracy and security, and demonstrates their privacy-preserving properties using the simulation paradigm. Section 4 compares the protocols' computational and communication complexities with the existing solutions, and Section 5 concludes this work.

2. Preliminary

2.1. Secure Multiparty Computation Model

2.1.1. Secure Two-Party Computation. A two-party computation is a random computation process, denoted by

$$f : \{0,1\}^* \times \{0,1\}^* \longrightarrow \{0,1\}^* \times \{0,1\}^*, \tag{1}$$

where

$$f : (x, y) \longrightarrow (f_1(x, y), f_2(x, y)). \tag{2}$$

2.1.2. Secure n-Party Computation. The above two-party computation can be extended to n-party computation. That is,

$$f : (x_1, \ldots, x_n) \longrightarrow (f_1(x_1, \ldots, x_n), \ldots, f_n(x_1, \ldots, x_n)). \tag{3}$$

2.1.3. Ideal SMC Model. The ideal SMC model is the most secure SMC model. It needs a trusted third party (TTP), that will always tell the truth and never lies. If such a TTP exists, Alice (holding x) and Bob (holding y) can privately compute $f(x, y)$ as follows:

(1) Alice sends x to TTP.

(2) Bob sends y to TTP.

(3) TTP computes $f(x, y) = (f_1(x, y), f_2(x, y))$.

(4) TTP sends $f_1(x, y)$ to Alice and sends $f_2(x, y)$ to Bob.

Theoretically, this protocol can solve any SMC problem, but the TTP is not easily implemented in practice. Therefore, SMC protocols without needing a TTP must be developed.

2.2. Semihonest Party.

At present, SMC protocols are investigated either in the semihonest model or in the malicious model. A semihonest party truthfully follows the protocol but may retain all intermediate computation and try to deduce other parties' private inputs from the record. Studying SMC in the semihonest model is the basis of studying SMC in the malicious model. The semihonest model is not only an important methodology but also provides an appropriate model for many settings. In some settings, proving that a

protocol is secure in the semihonest model is sufficient. Furthermore, Goldreich et el. [2] have proven that, given a protocol that privately computes a function f in the semihonest model, a protocol that privately computes the function f in the malicious model can be automatically produced by introducing a bit commitment macro that forces each party to either behave in the semihonest manner or to be detected. Therefore, this work focuses on solutions in the semihonest model.

2.3. Simulation Paradigm. A protocol is considered secure if what a party can obtain from the execution of the protocol can also be computed only from his input and output. This situation is formalized by a simulation paradigm [9] in which a party's view can be simulated by the input and output of the protocol. In this case, the parties cannot learn additional information except the necessary results from the output.

The simulation paradigm is a commonly used and widely accepted proof method for secure multiparty computation. The principle of the simulation paradigm is that the security of an SMC protocol is compared to the security of the ideal SMC protocol, and the protocol is secure if it discloses no more information than the ideal SMC protocol does. Therefore, the simulation paradigm is considered as the formal expression of principle to evaluate the security of SMC protocols.

2.3.1. Simulation Paradigm for the Two-Party Case. Suppose that $f = (f_1, f_2)$ is a probabilistic polynomial-time function. Alice holds x, and Bob holds y. They want to privately compute $f(x, y)$. π is a protocol that computes f.

In the execution of π, Alice and Bob obtain message sequences $\text{view}_1^\pi(x, y) = (x, r^1, m_1^1, \ldots, m_t^1)$ and $\text{view}_2^\pi(x, y) = (y, r^2, m_1^2, \ldots, m_t^2)$, respectively, where r^1 (r^2) is the result of her (his) internal coin toss and m_i^1 (m_i^2) is her (his) received message. Their outputs are $\text{output}_1^\pi(x, y)$ and $\text{output}_2^\pi(x, y)$.

Definition 1 (see [9]). For a function f, π privately computes f if there exist probabilistic polynomial-time algorithms S_1 and S_2 such that

$$\begin{aligned} &\left(S_1(x, f_1(x, y)), f_2(x, y)\right) \\ &\quad \stackrel{c}{\equiv} \left(\text{view}_1^\pi(x, y), \text{output}_2^\pi(x, y)\right), \\ &\left(f_1(x, y), S_2(y, f_2(x, y))\right) \\ &\quad \stackrel{c}{\equiv} \left(\text{output}_1^\pi(x, y), \text{view}_2^\pi(x, y)\right), \end{aligned} \tag{4}$$

where $\stackrel{c}{\equiv}$ denotes computational indistinguishability.

In order to prove that a multiparty computation protocol is secure, we must construct simulators S_1 and S_2 such that (4) hold.

2.3.2. Simulation Paradigm for the n-Party Case. A semihonest party is one that follows the protocol properly, with the exception that it keeps a record of all its intermediate computations. Loosely speaking, a multiparty protocol privately computes f if whatever a set I (or a coalition) of semihonest parties can obtain after participating in the protocol could be essentially obtained from the input and output of these parties. Thus, the only difference between the current definition and the one used in the two-party case is that we consider the gain of a coalition (rather than of a single party) from participating in the protocol [9].

Let $f : (\{0,1\}^*)^m \to (\{0,1\}^*)^m$ be an m-ary function, where $f_i(x_1, \ldots, x_m)$ denotes the i-th element of $f(x_1, \ldots, x_m)$. For $I = \{i_1, \ldots, i_t\} \subseteq [m] \stackrel{\text{def}}{=} \{1, \ldots, m\}$, we let $f_I(x_1, \ldots, x_m)$ denote the subsequence $f_{i_1}(x_1, \ldots, x_m), \ldots, f_{i_t}(x_1, \ldots, x_m)$. Let Π be an m-party protocol for computing f. The view of the i-th party during an execution of Π on $X = (x_1, \ldots, x_m)$, denoted by $\text{view}_i^\Pi(X)$, is defined as in Definition 1, and for $I = \{i_1, \ldots, i_t\}$, we let $\text{view}_I^\Pi(X) \stackrel{\text{def}}{=} (I, \text{view}_{i_1}^\Pi(X), \ldots, \text{view}_{i_t}^\Pi(X))$.

Definition 2 (see [9]). In case f is a deterministic m-ary function, we say that Π privately computes f if there exists a probabilistic polynomial-time algorithm, denoted by S, such that for every $I \subseteq [m]$, it holds that

$$\begin{aligned} &\left\{S\left(I, \left(x_{i_1}, \ldots, x_{i_t}\right), f_I(X)\right)\right\}_{X \in (\{0,1\}^*)^m} \\ &\quad \stackrel{c}{\equiv} \left\{\text{view}_I^\Pi(X)\right\}_{X \in (\{0,1\}^*)^m}. \end{aligned} \tag{5}$$

2.4. Paillier Encryption Scheme. The Paillier public-key cryptosystem [34] is probabilistic and has the additive homomorphism. The cryptosystem contains the following three algorithms.

KeyGen. This algorithm generates two large primes p, q and sets $N = pq$. Let $\lambda = \text{lcm}(p-1, q-1)$. The algorithm computes g such that $\gcd(L(g^\lambda \bmod N^2), N) = 1$, where

$$L(x) = \frac{x-1}{N}. \tag{6}$$

The public key is (g, N), and the private key is λ.

Encryption. To encrypt a plaintext m ($m < N$), select a random number r ($r < N$) and compute

$$c = g^m \cdot r^N \bmod N^2. \tag{7}$$

Decryption. To decrypt the ciphertext $c < N^2$, compute

$$m = \frac{L\left(c^\lambda \bmod N^2\right)}{L\left(g^\lambda \bmod N^2\right)} \bmod N. \tag{8}$$

Additive Homomorphism. Consider the following:

$$\begin{aligned} &E(m_1) \cdot E(m_2) \\ &\quad \equiv \left(g^{m_1} \cdot r_1^N \bmod N^2\right)\left(g^{m_2} \cdot r_2^N \bmod N^2\right) \\ &\quad = g^{(m_1+m_2)} \cdot (r_1 r_2)^N \bmod N^2 \equiv E(m_1 + m_2). \end{aligned} \tag{9}$$

2.5. Vectorization Method. We use a vectorization method to solve the GT problem privately. The vectorization method encodes a number into a vector. Suppose that $k \in U = \{u_1, \ldots, u_s\}$, k is encoded into an s-dimensional vector $V = (v_1, \ldots, v_s)$ as follows:

$$v_i = \begin{cases} \alpha, & u_i < k, \\ \gamma, & u_i = k, \\ \beta, & u_i > k. \end{cases} \tag{10}$$

The computational complexity of our protocol depends on $|U|$. Note that $s = |U|$. If $|U|$ is smaller, no matter how large k is, it can be encoded into a low dimension vector. For example, suppose that $k = 8388608$, $\alpha = 2$, $\beta = 3$, $\gamma = 5$ and $U = \{107, 1587, 357862, 8178261, 8388608, 11587243, 654395824\}$; then, the vector representation of k is $V = (2, 2, 2, 2, 5, 3, 3)$. The vector dimension is 7, which is much less than the bit number of $k \log_2 8388608 = 23$. With this vectorization method and the additive homomorphism of the Paillier encryption scheme, we can easily solve the GT problem.

Yao is the pioneer of secure multiparty computation research. In his seminal paper entitled "Protocols for Secure Computations" [1] which introduced the famous millionaire's problem and further ignited the secure multiparty computation research, he assumed that the numbers to be compared are members of a set $U = \{1, \ldots, 10\}$. Noting that the domain size $|U|$ is usually small in many practical applications [35], this assumption is reasonable and it does not leak any information.

In many cases, U is known to all parties in reality. For example, if some students want to know their score sorting, then $U = \{0, \ldots, 100\}$ is known to all parties; if two persons want to compare their ages, then U may be $\{1, \ldots, 120\}$; if two workers of a company want to compare their wages, U may be $\{1000, \ldots, 100000\}$, but the wages of a company are often sparsely rather than densely distributed on U. The wages can only be a few scales, and $|U|$ is small. Therefore, U is known to them and, in all these cases, U does not leak any information about their private data. Generally speaking, all the numbers compared in secure multiparty computation are comparable. If the numbers are comparable, the parties know their range. A common worker will never compare his wages with that of Bill Gates because they are not comparable. But we have to say that though U is known, if $|U|$ is large, the computational complexity of the protocol will become high, and the protocol therefore becomes impractical.

3. Our Protocols

In this section, we propose protocols to solve secure two-party and multiparty GT problems. In the two-party case, we propose Protocol 1 by using the vectorization method to encode a plaintext number into a vector and using the Paillier encryption scheme to encrypt the vector's components. Protocol 1 can determine the relationship of $x > y$, $x < y$ or $x = y$ in one execution.

Although we can expand the two-party GT protocol to n parties, the communication complexity is high, and it will disclose more information. Therefore, we propose Protocol 2 by using the vectorization method and the secret splitting method to solve the multiparty GT problem without using any encryption scheme based on computational assumptions. Protocol 2 can resist collusion attacks and is information-theoretically secure.

3.1. Secure Two-Party Computation for the GT Problem

3.1.1. A Protocol for the Two-Party GT Problem. Yao's scheme [1] simply determines whether $x \leq y$ or $x > y$, which cannot distinguish $x < y$ from $x = y$. Our protocol can compare $x > y$, $x < y$, and $x = y$ in one execution.

In our solution, Alice (holding x) uses the vectorization method to encode x into a vector and uses the Paillier encryption scheme to encrypt the vector. Bob (holding y) receives the encrypted vector, selects the y-th element, encrypts the y-th element into another ciphertext using the additive homomorphism of the Paillier encryption scheme, and returns the ciphertext to Alice. Alice decrypts the ciphertext and tells Bob the result.

The solution for two-party GT problem is as follows.

Protocol 1. Secure two-party computation of the GT problem.

Input. Alice's input is x, and Bob's input is y. So $x, y \in U = \{u_1, \ldots, u_s\}$.

Output. The relationship of x and y is denoted by

$$P(x, y) = \begin{cases} -1, & x > y, \\ 0, & x = y, \\ 1, & x < y. \end{cases} \tag{11}$$

(1) With the Paillier encryption scheme, Alice generates the public key $\{g, N\}$ and private key $\{\lambda\}$ and sends the public key to Bob.

(2) Following the vectorization method, Alice encodes x into a vector:

$$X = (m_1, \ldots, m_i, \ldots, m_s), \tag{12}$$

where

$$m_i = \begin{cases} \alpha, & 1 \leq u_i < x \\ \gamma, & u_i = x \\ \beta, & u_i > x \end{cases} \quad (\alpha \neq \beta \neq \gamma). \tag{13}$$

(3) Alice selects s random numbers $r_1, \ldots, r_s$, and encrypts the vector X using the Paillier encryption scheme:

$$E(X) = (E(m_1, r_1), \ldots, E(m_i, r_i), \ldots, E(m_s, r_s)), \tag{14}$$

where $E(m_i, r_i) = g^{m_i} \cdot r_i^N \bmod N^2$ $(i = 1, \ldots, s)$.

(4) Alice sends $E(X)$ to Bob.

(5) Bob selects a random number r_b. He chooses $E(m_i, r_i)$ from $E(X)$ such that $i = y$ and computes

$$E(m_i, r_i) \times E(0, r_b) = g^{m_i} \cdot r_i^n \bmod n^2 \times \left(g^0 \cdot r_b^n \bmod n^2\right) \longrightarrow E(\mu). \quad (15)$$

(6) Bob sends $E(\mu)$ to Alice.

(7) Alice decrypts

$$D(E(\mu)) = \mu \quad (16)$$

and tells Bob the result $P(x, y)$:

If $\mu = \alpha$, $P(x, y) = -1$ and $x > y$.

If $\mu = \beta$, $P(x, y) = 1$ and $x < y$.

If $\mu = \gamma$, $P(x, y) = 0$ and $x = y$.

3.1.2. Accuracy and Security

(1) Alice computes $D(E(\mu)) = \mu$; that is, $\mu = m_i$ such that $i = y$. Since $m_i \in \{m_1, \ldots, m_{x-1}, m_x, m_{x+1}, \ldots, m_s\} = \{\alpha, \ldots, \alpha, \gamma, \beta, \ldots, \beta\}$, if $m_i = \alpha$ which implies $m_i \in \{m_1, \ldots, m_{x-1}\}$, so $y < x$. If $m_i = \gamma$ which implies $m_i = m_x$, then $y = x$. If $m_i = \beta$ which implies $m_i \in \{m_{x+1}, \ldots, m_s\}$, then $y > x$.

(2) When Alice receives $E(\mu)$ from Bob, she does not know how to compute $E(\mu)$ because she does not know r_b, so $E(m_i, r_i)$ is private for Bob. Therefore, y is private for Bob.

When Bob obtains the result $x > y$ or $x < y$, he cannot know which m_i equals α or β, so x is private for Alice.

(3) Using Protocol 1, we can determine the relationship between x and y in one execution, especially $x = y$ that Yao's solution [1] cannot determine directly.

The following theorem refers to the privacy-preserving property of this protocol.

Theorem 3. *Protocol 1 for the two-party GT problem is secure in the semihonest model.*

Proof. We begin by constructing S_1 and S_2 such that (4) are satisfied.

In the protocol,

$$\text{view}_1^{\pi}(x, y) = \{x, E(X), E(\mu), \mu, P(x, y)\},$$
$$f_1(x, y) = f_2(x, y) = \text{output}_1^{\pi}(x, y) = \text{output}_2^{\pi}(x, y) = P(x, y), \quad (17)$$

where x, y are inputs, $E(X)$ is Alice's encryption result, μ is Alice's decryption result, $E(\mu)$ is Bob's computation result, and $P(x, y)$ is the output.

S_1 proceeds as follows:

(1) By $f_1(x, y)$, S_1 randomly selects a number $y' \in U$ such that $f_1(x, y) = f_1(x, y')$. S_1 constructs the vector $X = (m_1, m_2, \ldots, m_s)$.

(2) Using the Paillier encryption scheme, S_1 encrypts X using different random numbers r_i:

$$E(X) = \left(E(m_1, r_1), E(m_2, r_2), \ldots, E(m_s, r_s)\right). \quad (18)$$

(3) S_1 selects a random number r' and computes $E(m_i, r_i) \times (r'^n \bmod n^2) \to E(\mu')$, $(i = y')$.

(4) S_1 decrypts $D(E(\mu')) = \mu'$. By comparing μ' with $\{\alpha, \beta, \gamma\}$, S_1 determines $P(x, y')$.

Let

$$\{S_1(x, P(x, y))\} = \left\{x, E(X), E\left(\mu'\right), \mu', P\left(x, y'\right)\right\}. \quad (19)$$

Since $E(\mu) \stackrel{c}{\equiv} E(\mu')$, $\mu \stackrel{c}{\equiv} \mu'$, and $P(x, y) = P(x, y')$, then

$$\{(S_1(x, f_1(x, y)), f_2(x, y))\}_{x,y} \stackrel{c}{\equiv} \{(\text{view}_1^{\pi}(x, y), \text{output}_2^{\pi}(x, y))\}_{x,y}. \quad (20)$$

Similarly, we can construct S_2, such that

$$\{(f_1(x, y), S_2(y, f_2(x, y)))\}_{x,y} \stackrel{c}{\equiv} \{(\text{output}_1^{\pi}(x, y), \text{view}_2^{\pi}(x, y))\}_{x,y}. \quad (21)$$

□

3.2. Secure Multiparty Computation Protocol for the GT Problem (Secure Sorting Problem). In practice, more than two parties want to privately compare all of their numbers. For example, some companies want to compare their turnovers, but they do not want to disclose their data, so they prefer a secure protocol that can sort their turnovers. As another example, students desire to know their own score's rank without publishing their scores, and thus they must privately determine the order of their own score.

To solve the secure multiparty GT problem (i.e., secure sorting problem), we may use Protocol 1 to compare pairwise, but this process is of high complexity and will disclose too much information to other parties. Therefore, we propose a more efficient protocol as follows.

3.2.1. Secure Sorting Problem Model. The set is defined as $K = \{(P_1, a_1), (P_2, a_2), \ldots, (P_z, a_z)\}$, where P_i $(i = 1, \ldots, z$, and z is the number of the parties) holds his own number a_i, $a_i \neq a_j$ $(i \neq j)$.

Consider $L = \{\text{ord}(a_1), \ldots, \text{ord}(a_z)\}$, where $\text{ord}(a_i)$ is the order of a_i.

$$F : K \longrightarrow L. \quad (22)$$

A protocol is proposed to compute F privately. In the protocol, P_i only knows $\text{ord}(a_i)$ and cannot know other parties' orders and any other information.

Table 1: Four-party vectorization method and sorting order.

$P_1 : a_1 = 4, V_1 \rightarrow$	0	0	0	α	α	α	α	α	$\cdots$
$P_2 : a_2 = 2, V_2 \rightarrow$	0	α	α	α	α	α	α	α	$\cdots$
$P_3 : a_3 = 6, V_3 \rightarrow$	0	0	0	0	0	α	α	α	$\cdots$
$P_4 : a_4 = 7, V_4 \rightarrow$	0	0	0	0	0	0	α	α	$\cdots$
Addition		α		2α		3α	4α		
Sorting order		$\text{ord}(a_2) = 1$		$\text{ord}(a_1) = 2$		$\text{ord}(a_3) = 3$	$\text{ord}(a_4) = 4$		

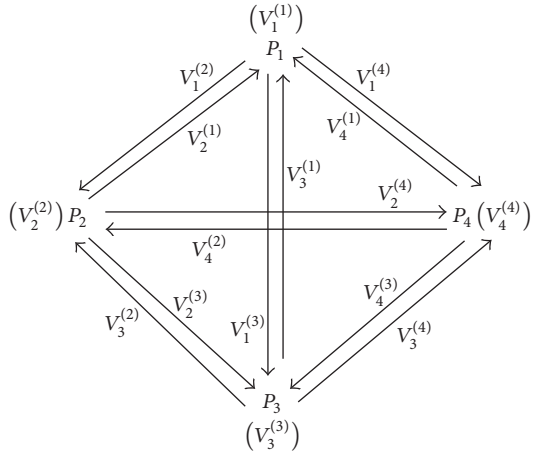

Figure 1: Four-party sending $V_i^{(k)}$.

3.2.2. A Protocol for the Multiparty GT Problem. In our protocol, $a_i \in U = \{u_1, \ldots, u_k, \ldots, u_s\}$, P_i encodes his own number a_i into a vector V_i by the vectorization method as follows.

$$a_i \longrightarrow V_i = (v_{i1}, \ldots, v_{ik}, \ldots, v_{is}), \tag{23}$$

where

$$v_{ik} = \begin{cases} 0, & 1 \le u_k < a_i, \\ \alpha, & u_k \ge a_i. \end{cases} \tag{24}$$

All parties' V_i form a matrix (Table 1). If P_i wants to know his order, he can add the components of a_ith column vector. For example, Table 1 shows the vectorization method and all parties' orders.

However, if P_i sends V_i to other parties directly, a_i will be disclosed. We utilize the secret splitting method to guarantee his privacy. P_i splits V_i into z vectors $V_i^{(1)}, \ldots, V_i^{(z)}$. He keeps $V_i^{(i)}$ secret and sends $V_i^{(1)}, \ldots, V_i^{(i-1)}, V_i^{(i+1)}, \ldots, V_i^{(z)}$ to other $z-1$ parties. In the end, P_i will receive $(z-1)$ vectors $V_j^{(i)}$ $(j \ne i)$ that other parties send to him, as shown in Figure 1. P_i computes $V_i' = V_i^{(i)} + \sum_{j=1}^{z} V_j^{(i)}$, $(j \ne i)$.

All parties publish their V_i'. If P_i wants to know his order, he adds all v'_{ja_i} of V_j', that is, $\sum_{j=1}^{z} v'_{ja_i} = \delta$, and $\text{ord}(a_i) = \delta/\alpha$.

Protocol 2. Secure multiparty computation of the GT problem.

Input. The inputs of $P_1, P_2, \ldots, P_z$ are $a_1, a_2, \ldots, a_z, a_i \in U = \{u_1, u_2, \ldots, u_s\}$.

Output. Consider $\text{ord}(a_1), \text{ord}(a_2), \ldots, \text{ord}(a_z)$.

(1) All parties encode their own numbers with the vectorization method. Take a_i for example:

$$a_i \longrightarrow V_i = (v_{i1}, \ldots, v_{ik}, \ldots, v_{is}), \tag{25}$$

where

$$v_{ik} = \begin{cases} 0, & 1 \le u_i < a_i, \\ \alpha, & k \ge a_i. \end{cases} \tag{26}$$

(2) P_i randomly splits V_i into z vectors $V_i^{(1)}, V_i^{(2)}, \ldots, V_i^{(z)}$, as follows:

$$V_i = (v_{i1}, v_{i2}, v_{i3}, \ldots, v_{ik}, \ldots, v_{is}) \longrightarrow$$
$$\begin{cases} V_i^{(1)} = \left(v_{i1}^{(1)}, v_{i2}^{(1)}, v_{i3}^{(1)}, \ldots, v_{ik}^{(1)}, \ldots, v_{is}^{(1)}\right), \\ V_i^{(2)} = \left(v_{i1}^{(2)}, v_{i2}^{(2)}, v_{i3}^{(2)}, \ldots, v_{ik}^{(2)}, \ldots, v_{is}^{(2)}\right), \\ \qquad \vdots \\ V_i^{(z)} = \left(v_{i1}^{(z)}, v_{i2}^{(z)}, v_{i3}^{(z)}, \ldots, v_{ik}^{(z)}, \ldots, v_{is}^{(z)}\right), \end{cases} \tag{27}$$

where all $v_{ik}^{(j)}$ $(1 \le j \le z, 1 \le k \le s)$ are random numbers that satisfy

$$v_{i1} = \sum_{j=1}^{z} v_{i1}^{(j)}, \ldots, v_{is} = \sum_{j=1}^{z} v_{is}^{(j)}. \tag{28}$$

(3) P_i keeps his $V_i^{(i)}$ secret and sends $V_i^{(j)}$ $(j \neq i)$ to other $z-1$ parties separately. In the end, each party holds z vectors; that is, P_i holds $V_1^{(i)}, \ldots, V_i^{(i)}, \ldots, V_z^{(i)}$ and adds the corresponding column vectors, denoted by $V_i' = \{v_{i1}', v_{i2}', \ldots, v_{is}'\} = \{(v_{11}^{(i)} + v_{21}^{(i)} + \cdots + v_{z1}^{(i)}), (v_{12}^{(i)} + v_{22}^{(i)} + \cdots + v_{z2}^{(i)}), \ldots, (v_{1s}^{(i)} + v_{2s}^{(i)} + \cdots + v_{zs}^{(i)})\}$.

(4) All parties publish their V_i'. According to a_i, P_i adds all v_{ja_i}' $(j = 1, \ldots, z)$ of V_j', denoted by $\delta_i = (v_{1a_i}' + v_{2a_i}' + v_{3a_i}' + \cdots + v_{za_i}')$.

(5) P_i computes $\mathrm{ord}(a_i) = \delta_i / \alpha$ as his order.

3.2.3. Accuracy and Security

(1) In this protocol, we use the secret splitting method to split a vector V into z vectors. Each party keeps his $V_i^{(i)}$ secret. Without $V_i^{(i)}$, others cannot conspire to compute V_i.

(2) Protocol 2 does not require any encryption scheme based on computational assumptions, so it is information-theoretically secure.

(3) P_i adds the a_i-th column vectors of V_j' $(j = 1, \ldots, z)$, and other parties do not know which column vector P_i selects.

(4) In a particular case, if P_i holds the number 2 and his order is also 2, he confirms that someone holds the number 1, but he does not know who holds the number 1. This situation does not imply that the protocol is insecure because this information is what the function computed leaks. Even in the ideal SMC protocol, the TTP tells P_i that his rank is 2, the information also leaks because this is what the function to be computed leaks.

(5) The protocol cannot be applied to the two-party situation, because, in this case,

in the execution of Protocol 2 for two parties, Alice splits V_1 to vectors $\{V_1^{(1)}, V_1^{(2)}\}$, and Bob splits V_2 to vectors $\{V_2^{(1)}, V_2^{(2)}\}$. Alice sends $V_1^{(2)}$ to Bob, and Bob sends $V_2^{(1)}$ to Alice. Bob computes $V_2' = V_2^{(2)} + V_1^{(2)}$ and sends V_2' to Alice. Because Alice has the vectors $V_1^{(2)}$ and $V_2^{(1)}$, she can compute $V_2^{(2)} = V_2' - V_1^{(2)}$ and compute $V_2 = V_2^{(1)} + V_2^{(2)}$, so she obtains Bob's a_2. Therefore, Protocol 2 is not applicable to the two-party GT problem.

(6) In this protocol, we do not consider attacks in the transmitting process. If necessary, we can use the Paillier encryption scheme [34] to encrypt $V_i^{(k)}$ and V_i', although this increases the computational and communication costs.

(7) In fact, in many cases, limited parties take part in sorting their numbers. In cryptographic literatures, the multiparty refers 3 to 5 parties. There are few literatures that consider more than 10 parties. In addition, in our life, we sort 100 richest men of the world or compare the top 100 banks in the world or rank the top 500 companies all over the world at most. Therefore, the number of parties in the above cases does matter much.

(8) Protocol 2 can resist collusion attacks, which is proved by the following Theorem 4.

The following theorem states the privacy-preserving property of the protocol.

Theorem 4. *Protocol 2 for the multiparty GT problem is secure in the semihonest model.*

Proof. In Protocol 2, even though some adversaries obtain P_i's $z-1$ vectors $V_j^{(i)}$ $(j \neq i)$, they cannot compute V_i. Because all vectors $V_j^{(i)}$ $(j = 1, \ldots, z)$ sum to V_i, while P_i keeps $V_i^{(i)}$ secret, the adversaries cannot obtain $V_i^{(i)}$. Therefore, the security of Protocol 2 is the same as the security of a protocol with a TTP in the ideal SMC model. We prove the privacy-preserving property of Protocol 2 using the following formal proving method (Definition 2).

We consider the following cases.

Case 1. P_1 does not participate in collusion, while other participants I $(I \subseteq \{P_2, P_3, \ldots, P_z\})$ conspire to compute P_j's V_j $(P_j \notin I)$. By executing Protocol 2, the set I can obtain $\{V_2', \ldots, V_{j-1}', V_{j+1}', \ldots, V_z'\}$ and $\{\mathrm{ord}(a_2), \ldots, \mathrm{ord}(a_{j-1}), \mathrm{ord}(a_{j+1}), \ldots, \mathrm{ord}(a_z)\}$. Since $P_1, P_j \notin I$, $V_1^{(1)}$ and $V_j^{(j)}$ are private, the coconspirators I cannot obtain V_1 and V_j. So they do not conspire a_j. We can construct a simulator S such that (5) holds.

(i) By the result of the protocol, the coconspirators I choose $V_1^{(1)*}$ and $V_j^{(j)*}$ such that

$$\begin{aligned}&\left(V_1^{(1)*} + V_1^{(2)} + \cdots + V_1^{(z)}\right) + V_2 + \cdots + V_{j-1} \\ &\quad + \left(V_j^{(j)*} + V_j^{(1)} + \cdots + V_j^{(z)}\right) + V_{j+1} + \cdots + V_z \\ &= V.\end{aligned} \tag{29}$$

(ii) The set I will execute the protocol Π, determining their view_I^{Π} as follows:

$$\begin{aligned}&\mathrm{view}_I^{\Pi}(V) = \left\{I, \mathrm{view}_2^{\Pi}(V), \ldots, \mathrm{view}_{j-1}^{\Pi}(V), \right. \\ &\left. \mathrm{view}_{j+1}^{\Pi}(V), \ldots, \mathrm{view}_z^{\Pi}(V)\right\} = \left\{V_1'^*, V_2', \ldots, V_{j-1}', V_j'^*, \right. \\ &\left. V_{j+1}', \ldots, V_z', V\right\}.\end{aligned} \tag{30}$$

(iii) Let

$$\begin{aligned}&S\left(I, \left(V_2', V_3', \ldots, V_z'\right), f(V)\right) \\ &\quad = \left\{V_1', V_2', \ldots, V_{j-1}', V_j', V_{j+1}', \ldots, V_z', V\right\}. \\ &V_1'^* \overset{c}{\equiv} V_1', \\ &V_j'^* \overset{c}{\equiv} V_j', \\ &\left\{\left(S\left(I, \left(V_2', V_3', \ldots, V_z'\right), f(V)\right)\right)\right\} \overset{c}{\equiv} \left\{\mathrm{view}_I^{\Pi}(V)\right\}.\end{aligned} \tag{31}$$

Therefore, in this case, the protocol is secure.

Case 2. P_1 and the participants $I \subseteq \{P_2, P_3, \ldots, P_z\}$ conspire to obtain the information of other participants $I' = \{P_1, P_2, \ldots, P_z\} \setminus I$.

Case 2.1. If $|I'| = 1$, suppose $P_i \in I'$, P_i's order ord(a_i) except a_i can be obtained, which is the same as the case mentioned above in part (4) and part (5) of Section 3.2.3. Even in the ideal SMC protocol, the TTP tells the orders of P_1 and I, and P_1 and I can conspire P_i's order ord(a_i). Therefore, the protocol does not leak more information than that the ideal SMC model leaks, and the protocol is secure.

Case 2.2. If $I' = \{P_i, P_j\}$, $V_i^{(i)}$ and $V_j^{(j)}$ are private, and P_1 and I cannot know V_i' and V_j', which is the same as Case 1. In this case, the protocol is secure. This is true for $|I'| \geqslant 2$.

To sum up, in any case, there exists a simulator S such that (5) holds, so Protocol 2 is secure. □

To help the readers understand the protocol, we give a toy example.

Example

Input. $\{(P_1, 5), (P_2, 3), (P_3, 8), (P_4, 7)\}$, $U = \{1, \ldots, 10\}$.

Output. ord(i), $i \in \{5, 3, 8, 7\}$.

(1) $P_1, \ldots, P_4$ encode their number's vectors as follows (set $\alpha = 2$):

$$\begin{aligned} x_1 &= 5, \\ V_1 &= (0, 0, 0, 0, 2, 2, 2, 2, 2, 2), \\ x_2 &= 3, \\ V_2 &= (0, 0, 2, 2, 2, 2, 2, 2, 2, 2), \\ x_3 &= 8, \\ V_3 &= (0, 0, 0, 0, 0, 0, 0, 2, 2, 2), \\ x_4 &= 7, \\ V_4 &= (0, 0, 0, 0, 0, 0, 2, 2, 2, 2). \end{aligned} \tag{32}$$

(2) $P_1, \ldots, P_4$ split their own vector into four random vectors, respectively:

$$V_1 \longrightarrow$$

$$\begin{cases} V_1^{(1)} = (-5, -3, 2, 3, 3, -1, -4, 2, 3, 5) \\ V_1^{(2)} = (7, 4, -5, 6, 2, -4, 9, -2, 1, 3) \\ V_1^{(3)} = (2, -1, 3, -2, 4, 5, 4, 1, 2, -3) \\ V_1^{(4)} = (-4, 0, 0, -7, -7, 2, -7, 1, -4, -3). \end{cases}$$

$$V_2 \longrightarrow$$

$$\begin{cases} V_2^{(1)} = (4, 2, 5, -7, -3, -5, 6, 1, 0, 1) \\ V_2^{(2)} = (-2, 3, -1, 6, 1, 2, -4, -5, -7, 3) \\ V_2^{(3)} = (3, -6, 2, 3, 2, 5, -3, -2, 4, -2) \\ V_2^{(4)} = (-5, 1, -4, 0, 2, 0, 3, 8, 5, 0). \end{cases}$$

$$V_3 \longrightarrow$$

$$\begin{cases} V_3^{(1)} = (2, 1, -3, -5, -2, 2, 0, 0, 6, -1) \\ V_3^{(2)} = (3, 0, -2, 0, -5, 0, -4, 0, -2, 2) \\ V_3^{(3)} = (-3, 2, 5, -4, 1, 0, 2, -1, -3, -2) \\ V_3^{(4)} = (-2, -3, 0, 9, 6, -2, 2, 3, 1, 3). \end{cases}$$

$$V_4 \longrightarrow$$

$$\begin{cases} V_4^{(1)} = (3, 1, -2, -5, -3, -2, -1, 0, -3, -4) \\ V_4^{(2)} = (2, 3, -1, 5, 7, 4, 2, 0, -3, 2) \\ V_4^{(3)} = (-4, 3, -2, -1, -5, 2, 1, 2, 3, 1) \\ V_4^{(4)} = (-1, -7, 5, 1, 1, -4, 0, 0, 5, 3). \end{cases} \tag{33}$$

(3) P_1 keeps $V_1^{(1)}$ secret and sends $V_1^{(2)}, V_1^{(3)}, V_1^{(4)}$ to P_2, P_3, P_4, respectively.

P_2 keeps $V_2^{(2)}$ secret and sends $V_2^{(1)}, V_2^{(3)}, V_2^{(4)}$ to P_1, P_3, P_4, respectively.

P_3 keeps $V_3^{(3)}$ secret and sends $V_3^{(1)}, V_3^{(2)}, V_3^{(4)}$ to P_1, P_2, P_4, respectively.

P_4 keeps $V_4^{(4)}$ secret and sends $V_4^{(1)}, V_4^{(2)}, V_4^{(3)}$ to P_1, P_2, P_3, respectively.

(4) P_1 holds $V_1^{(1)}, V_2^{(1)}, V_3^{(1)}, V_4^{(1)}$.

P_2 holds $V_2^{(2)}, V_1^{(2)}, V_3^{(2)}, V_4^{(2)}$.

P_3 holds $V_3^{(3)}, V_1^{(3)}, V_2^{(3)}, V_4^{(3)}$.

P_4 holds $V_4^{(4)}, V_1^{(4)}, V_2^{(4)}, V_3^{(4)}$.

(5) P_1 computes $V_1' = V_1^{(1)} + V_2^{(1)} + V_3^{(1)} + V_4^{(1)} = (4, 1, 2, -14, -5, -6, 1, 3, 6, 1)$.

P_2 computes $V_2' = V_2^{(2)} + V_1^{(2)} + V_3^{(2)} + V_4^{(2)} = (10, 10, -9, 17, 5, 2, 3, -7, -11, 10)$.

P_3 computes $V_3' = V_3^{(3)} + V_1^{(3)} + V_2^{(3)} + V_4^{(3)} = (-2, -2, 8, -4, 2, 12, 4, 0, 6, -6)$.

P_4 computes $V_4' = V_4^{(4)} + V_1^{(4)} + V_2^{(4)} + V_3^{(4)} = (-12, -9, 1, 3, 2, -4, -2, 12, 7, 3)$.

(6) P_1, P_2, P_3, P_4 publish their V_1', V_2', V_3', V_4', respectively.

(7) By $a_1 = 5$, P_1 selects $V_1'(5) = -5$, $V_2'(5) = 5$, $V_3'(5) = 2$, $V_4'(5) = 2$.

By $a_2 = 3$, P_2 selects $V_1'(3) = 2$, $V_2'(3) = -9$, $V_3'(3) = 8$, $V_4'(3) = 1$.

By $a_3 = 8$, P_3 selects $V_1'(8) = 3$, $V_2'(8) = -7$, $V_3'(8) = 0$, $V_4'(8) = 12$.

By $a_1 = 3$, P_4 selects $V_1'(7) = 1$, $V_2'(7) = 3$, $V_3'(7) = 4$, $V_4'(7) = -2$.

(8) P_1 computes his rank ord$(5) = (V_1'(5) + V_2'(5) + V_3'(5) + V_4'(5))/\alpha = (-5 + 5 + 2 + 2)/2 = 2$.

Table 2: Computational complexities for the two-party GT problem.

Schemes	Result	Modular multiplications
Yao [1]	$>, \leq$	Exponential
Blake and Kolesnikov [25]	$>, \leq$	$(4b + 1) \lg N + 6b$
Lin and Tzeng [26]	$>, \leq$	$5b \lg N + 4b - 6$
Protocol 1	$>, <, =$	$2(s + 2) \lg N$

b: bit number of inputs, N: modulus of the public-key encryption scheme, and s: encoding vector dimension.

P_2 computes his rank $\text{ord}(3) = (V_1'(3)+V_2'(3)+V_3'(3)+V_4'(3))/\alpha = (2 + (-9) + 8 + 1)/2 = 1$.

P_3 computes his rank $\text{ord}(8) = (V_1'(8)+V_2'(8)+V_3'(8)+V_4'(8))/\alpha = (3 - 7 + 0 + 12)/2 = 4$.

P_4 computes his rank $\text{ord}(7) = (V_1'(7)+V_2'(7)+V_3'(7)+V_4'(7))/\alpha = (1 + 3 + 4 + (-2))/2 = 3$.

4. Computational and Communication Complexity

4.1. Computational Complexity. The computational complexity is an important measure for evaluating the efficiency of an SMC protocol.

For secure two-party computation of the GT problem, we compare the computational complexity of Protocol 1 with that of the following typical solutions. The computational cost of [1] is exponential, which is impractical if the two numbers are very large. Furthermore, the protocol only compares $x > y$ or $x \leq y$. In [25], which also computes only $x > y$ or $x \leq y$, the computational cost is $((4b + 1) \lg N + 6b)$ modular multiplications, where b is the bit number of inputs and N is the modulus of the Paillier encryption scheme. In [26], Lin and Tzeng used the multiplicative homomorphism of the ElGamal encryption scheme to solve the GT problem for $x > y$ or $x \leq y$, the computational cost of which is $(5b \lg N + 4b - 6)$ modular multiplications. In addition, some researchers proposed SMC protocols for the GT problem based on linear secret sharing. Nishide and Ohta [29] constructed a simplified bit-decomposition protocol for the GT problem, which cannot determine $x = y$.

In the proposed Protocol 1, Alice encodes her number x into an s-dimensional vector. She encrypts s components of the encoding vector and computes s-time Paillier encryption and one-time Paillier decryption. Bob computes one-time Paillier encryption. The computational cost of each Paillier encryption or decryption is $2 \lg N$ modular multiplications. Therefore, the computational overhead of Protocol 1 is $(2(s + 2) \lg N)$ modular multiplications, where s is the dimension of the encoding vector and N is the modulus of the Paillier encryption scheme. Table 2 compares the computational complexities of the proposed protocol and the existing protocols for the two-party GT problem.

Table 2 shows that Protocol 1 can determine whether $x > y$, $x < y$ or $x = y$ in one execution, whereas the other three solutions can only determine whether $x > y$ or $x \leq y$. In addition, s is much smaller than b, because s is the dimension of the encoding vector, whereas b is the bit number of inputs. For example, $x = 8388608$ and $U = \{107, 1587, 357862, 8178261, 8388608, 11587243, 654395824\}$, $\alpha = 2$, $\beta = 3$, $\gamma = 5$; thus, its encoding vector is $v = (2, 2, 2, 2, 5, 3, 3)$, so $s = 7$. But the bit number of $x = 8388608$ is $b = \log_2 8388608 = 23$. Therefore, Protocol 1 is more efficient than these three existing solutions.

Table 3: Computational complexities for the n-party GT problem.

Schemes	Modular multiplications
Damgard et al. [27]	$l \log l$
Tang et al. [28]	$n^2 t(4t + 3)$
Protocol 2	Negligible

n: the number of parties, t: the threshold value, and l: the length of the shared values.

For secure multiparty computation of the GT problem (secure sorting problem), the solution of [1] is not suitable for the multiparty GT problem, and we compare the computational cost of Protocol 2 with that of [27, 28] which are typical protocols for the secure sorting problem. Paper [27] proposed an SMC protocol for the sorting problem based on the secret sharing scheme. The computational cost of [27] is $l \log l$ (l is the length of the shared values) modular multiplications. Paper [28] proposed a sorting protocol based on the secret sharing method, with a computational cost of $n^2 t(4t + 3)$ modular multiplications, where n is the number of parties and t is the threshold value.

In Protocol 2, we simply use the vectorization and the secret splitting methods, which requires $(2ns + 1)(n - 1)$ additions, where s is the encoding vector dimension and n is the number of parties. For example, when $n = 4$, $s = 10$, the computational cost is 243 additions, which is much smaller than that using any public-key encryption scheme. Therefore, the computational cost of Protocol 2 is negligible. Table 3 summarizes these computational complexities for the multiparty GT problem.

4.2. Communication Complexity. The communication complexity, measured by the number of communicating rounds, is another important measure for evaluating the efficiency of an interactive protocol. For secure two-party computation of the GT problem, [1] requires two rounds, and [25] requires four rounds. Nishide and Ohta [29] constructed a simplified bit-decomposition protocol for the GT problem, whose communication cost is 15 rounds, whereas Protocol 1 requires only one round.

Table 4: The communication complexities for the GT problem.

Schemes	Two-party	n-party
Yao [1]	2	—
Blake and Kolesnikov [25]	4	$2n(n-1)$
Nishide and Ohta [29]	15	—
Tang et al. [28]	—	$n(n+1)$
Protocol 1	1	—
Protocol 2	—	$2n(n-1)$

n: the number of parties.

For secure n-party computation of the GT problem, [25] requires $2n(n-1)$ rounds, and [28] requires $n(n+1)$ rounds. Protocol 2 also requires $2n(n-1)$ rounds.

Table 4 summarizes the communication complexities of these schemes for both types of the GT problem.

5. Conclusion

In this work, we propose the SMC protocols to solve the two-party and multiparty GT problem. Using the Paillier encryption scheme and the vectorization method, we construct an SMC protocol for solving the two-party GT problem which can determine $x > y$, $x < y$, $x = y$ in one execution. To efficiently solve the multiparty GT problem (secure sorting problem), we use the vectorization method to transform private numbers into vectors and use the secret splitting method to resist collusion attacks without any encryption scheme based on computational assumptions. This protocol is information-theoretically secure. The simulation paradigm proves the privacy-preserving property of the protocols in the semihonest model. Comparing the computational and communication complexities of our protocols with that of the existing solutions, our protocols are efficient for practical applications and easily implemented.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research is supported by the Natural Science Foundation of China (Grant nos. 61272435, 61272514, 61261028, and 61562065), the Natural Science Foundation of Inner Mongolia (Grant no. 2017MS0602), the University Scientific Research Project of Inner Mongolia (Grant no. NJZY17166), and Fundamental Research Funds for the Central Universities (Grant no. 2016TS061). The authors thank the sponsors for their support.

References

[1] A. Yao, "Protocols for secure computations," in *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, pp. 160–164, IEEE Computer Society Press, Los Alamitos, Calif, USA, 1982.

[2] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game," in *Proceeding of the nineteenth annual ACM conference on Theory of Computing*, pp. 218–229, IEEE Press, Piscataway, NJ, USA, 1987.

[3] S. Goldwasser, "Multi-party computations: past and present," in *Proceedings of the 16th Annual ACM Symposium on Principles of Distributed Computing*, pp. 1–6, ACM Press, New york, NY, USA, 1997.

[4] W. Liu, Y.-B. Wang, Z.-T. Jiang, and Y.-Z. Cao, "A protocol for the quantum private comparison of equality with x-type state," *International Journal of Theoretical Physics*, vol. 51, no. 1, pp. 69–77, 2012.

[5] D. Dachman-Soled, T. Malkin, M. Raykova, and M. Yung, "Efficient robust private set intersection," *International Journal of Applied Cryptography*, vol. 2, no. 4, pp. 289–303, 2012.

[6] S. Li, C. Wu, D. Wang, and Y. Dai, "Secure multiparty computation of solid geometric problems and their applications," *Information Sciences*, vol. 282, pp. 401–413, 2014.

[7] J. Alwen, R. Ostrovsky, H. S. Zhou et al., "Incoercible multiparty computation and universally composable receipt-free voting," in *Proceedings of the Annual Cryptology Conference*, pp. 763–780, Springer, Berlin, Heidelberg, Germany, 2015.

[8] D. Bogdanov, M. Niitsoo, T. Toft, and J. Willemson, "High-performance secure multi-party computation for data mining applications," *International Journal of Information Security*, vol. 11, no. 6, pp. 403–418, 2012.

[9] O. Goldreich, *The Fundamental of Cryptography: Basic Applications*, Cambridge University Press, London, UK, 2004.

[10] Z. Fu, X. Sun, Q. Liu, L. Zhou, and J. Shu, "Achieving efficient cloud search services: multi-keyword ranked search over encrypted cloud data supporting parallel computing," *IEICE Transactions on Communications*, vol. E98B, no. 1, pp. 190–200, 2015.

[11] Z. Xia, X. Wang, X. Sun, Q. Liu, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, pp. 340–352, 2015.

[12] Z. Pan, Y. Zhang, and S. Kwong, "Efficient motion and disparity estimation optimization for low complexity multiview video coding," *IEEE Transactions on Broadcasting*, vol. 61, no. 2, pp. 166–176, 2015.

[13] B. Gu, V. S. Sheng, K. Y. Tay, W. Romano, and S. Li, "Incremental support vector learning for ordinal regression," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 7, pp. 1403–1416, 2015.

[14] Z. H. Xia, X. H. Wang, L. G. Zhang, Z. Qin, X. M. Sun, and K. Ren, "A privacy-preserving and copy-deterrence content-based image retrieval scheme in cloud computing," *IEEE Transactions*

on Information Forensics and Security, vol. 11, pp. 2594–2608, 2016.

[15] Y. Zhang, X. Sun, and W. Wang, "Efficient algorithm for k-barrier coverage based on integer linear programming," *China Communications*, vol. 13, no. 7, pp. 16–23, 2016.

[16] Y. M. Guo, S. F. Zhou, J. W. Dou, S. D. Li, and D. S. Wang, "Efficient privacy-preserving interval computation and its applications," *Chinese Journal of Computers*, no. 39, pp. 1–17, 2016.

[17] S. D. Li, X. L. Yang, X. J. Zuo, and J. Kang, "Privacy protecting similitude determination for graphics similarity," *Chinese Journal of Electronics*, vol. 10, pp. 1–15, 2016 (Chinese).

[18] A. Zaman, M. A. Siddique, and Y. Morimoto, "Secure computation of skyline query in mapreduce," in *Proceedings of the Advanced Data Mining and Applications, 12th International Conference*, pp. 345–360, 2016.

[19] J. Doerner and D. Evans, "Secure stable matching at scale," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1602–1613, Vienna, Austria, 2016.

[20] Y. Ishai, E. Kushilevitz, S. Lu et al., "Private large-scale databases with distributed searchable symmetric encryption," in *Proceedings of the Cryptographers' Track at the RSA Conference*, pp. 90–107, Springer International Publishing, Berlin, Heidelberg, Germany, 2016.

[21] M. Chaudhari and J. Varmora, "Advance privacy preserving in association rule mining," in *The Proceedings of the International Conference on Electrical, Electronics, and Optimization*, pp. 2527–2530, 2016.

[22] D. Bogdanov, L. Kamm, S. Laur et al., "Rmind: a tool for cryptographically secure statistical analysis," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–14, 1016.

[23] A. Aly and M. Van Vyve, "Practically efficient secure single-commodity multi-market auctions," in *Financial Cryptography*, vol. 7859, pp. 1–18, LNCS, Springer, Heidelberg, Germany, 2016.

[24] M. Blanton and E. Aguiar, "Private and oblivious set and multiset operations," *International Journal of Information Security*, vol. 15, no. 5, pp. 493–518, 2016.

[25] I. F. Blake and V. Kolesnikov, "Strong conditional oblivious transfer and computing on intervals," in *Proceeding of the Advances in Cryptology-ASIACRYPT*, pp. 515–529, Springer, Berlin, Germany, 2004.

[26] H. Y. Lin and W. G. Tzeng, "An efficient solution to the millionaires' problem based on homomorphic encryption," in *Proceeding of the Applied Cryptography and Network Security Conference*, vol. 6, pp. 456–466, New York, NY, USA, 2005.

[27] I. Damgard, M. Fitzi, E. Kiltz, J. B. Nielsen, and T. Toft, "Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation," in *Proceeding of the Theory of Cryptography Conference*, vol. 3876 of *Lecture Notes in Computer Science*, pp. 285–304, Springer, Berlin, Germany, 2006.

[28] C. Tang, G. Shi, and Z. Yao, "Secure multi-party computation protocol for sequencing problem," *Science China Information Sciences*, vol. 54, no. 8, pp. 1654–1662, 2011.

[29] T. Nishide and K. Ohta, "Multiparty computation for interval, equality, and comparison without bit-decomposition protocol," in *Proceedings of the International Workshop on Public Key Cryptography*, vol. 4450, pp. 343–360, Springer, Berlin, Heidelberg, Germany, 2007.

[30] D. Grigoriev and V. Shpilrain, "Yao's millionaires' problem and decoy-based public key encryption by classical physics," *International Journal of Foundations of Computer Science*, vol. 25, no. 4, pp. 409–417, 2014.

[31] Y.-J. Chang, C.-W. Tsai, and T. Hwang, "Multi-user private comparison protocol using GHZ class states," *Quantum Information Processing*, vol. 12, no. 2, pp. 1077–1088, 2013.

[32] S. D. Li and D. S. Wang, "Efficient secure multiparty computation based on homomorphic encryption," *Chinese Journal of Electronics*, vol. 4, pp. 798–803, 2013 (Chinese).

[33] Z. Cao and L. Liu, Improvement of Lin-Tzeng solution to Yao's Millionaires problem and its cheating advantage analysis. IACR Cryptology ePrint Archive, 2013, 2013: 788.

[34] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology-EUROCRYPT*, pp. 223–238, Springer, Berlin, Germany.

[35] B. Samanthula and W. Jiang, "Secure multiset intersection cardinality and its application to jaccard coefficient," *IEEE Transactions on Dependable*, vol. 13, no. 5, pp. 591–604, 2016.