Received 26 April 2016; | Revised 6 June 2016; | Accepted 6 June 2016

DOI: xxx/xxxx

ARTICLE TYPE

Efficient Solution to the Millionaires' Problem Based on Asymmetric Commutative Encryption Scheme

Meng Liu*[1,2] | Yun Luo[3] | Priyadarsi Nanda[4] | Shui Yu[5]

[1]School of Mechanical, Electrical and Information Engineering, Shandong University, Weihai, Shandong, China
[2]Department of Electrical and Computer Engineering, University of Auckland, New Zealand
[3]Faculty of Computer Science and Technology, Guizhou University, Guizhou, China
[4]School of Electrical and Data Engineering, University of Technology Sydney, Australia
[5]School of Information Technology, Deakin University, Burwood, Australia

Correspondence
*Meng Liu, School of Mechanical, Electrical and Information Engineering, Shandong University, Weihai Email: liumeng@sdu.edu.cn

Summary

Secure multiparty computation (SMC) is an important scheme in cryptography and can be applied in various real-life problems. The first SMC problem is the millionaires' problem, and its protocol is an important building block. Because of the less efficiency of public key encryption scheme, most existing solutions based on public key cryptography to this problem is inefficient. Thus, a solution based on the symmetric encryption scheme has been proposed. In this paper, we formally analyse the vulnerability of this solution, and propose a new scheme based on the Decisional Diffie-Hellman (DDH) assumption. Our solution also uses 0-encoding and 1-encoding generated by our modified encoding method to reduce the computation cost. We implement the solution based on symmetric encryption scheme and our protocol. Extensive experiments are conducted to evaluate the efficiency of our solution, and the experimental results show that our solution can be much more efficient and be approximately 8000 times faster than the solution based on symmetric encryption scheme for a 32-bit input and short-term security. Moreover, our solution is also more efficient than the state-of-the-art solution without pre-computation and can also compare well with the state-of-the-art protocol while the bit length of private inputs is large enough.

KEYWORDS:
secure multiparty computation, millionaires' problem, commutative encryption, set-inclusion problem, security analysis

1 | INTRODUCTION

Secure multiparty computation (SMC) was first proposed by Yao in 1982[1]. The goal of SMC is to enable parties to jointly compute a function over their inputs without revealing these private inputs. For example, a given number of parities p_1, p_2, ..., p_n, all participants have a private input data, respectively d_1, d_2, ..., d_n. They want to compute the value of a public function f on n variables (d_1, d_2, ..., d_n). An SMC protocol is secure if no participant can learn more than what he/she can learn from his/her own input from the public function and the result. SMC appears as an essential problem in cryptography and its solutions have been utilized in cooperative scientific computation[2,3], data mining [4,5], privacy-preserving clustering[6], bidding and auction in e-commerce[7,8,9], secure computational geometry[10,11,12,13,14], set intersection[15,16,17], secure statistical analysis[18], privacy-preserving image retrieval[19,20,21,22] and secure data aggregation in Smart Metering systems[23].

The first SMC problem is Yao's Millionaires' problem. It is a secure two-party computations problem and it has served as an important building blocks in some solutions[10,19,20,24,25,26,27]. The problem discusses two millionaires, Alice and Bob, who want to know which of them is richer without disclosing their actual wealth. This problem is analogous to a more general problem where Alice and Bob have their private inputs, x and y, and they want to determine the predicate $x > y$ without revealing the actual values of x and y. The first solution to millionaires' problem is presented by Yao himself. For the n-bit numbers x and y, it needs 1 time public key encryption, 2^n times public key decryptions, 2^n times modular operation, and at least $2^{2n}/2$ times verifications. So it is exponential in time and space and too expensive to be practical.

Many other solutions have been proposed to solve Millionaires' problem. Ioannidis et al.[28] use 1-out-of-2 oblivious transfer scheme to construct a protocol that runs n times of the OT scheme, where n is the length of the private inputs. The implementation of the 1-out-of-2 oblivious transfer based on public key cryptography needs 4 times public key encryptions and 2 decryptions. Let $N = 2^n$ be the maximal value of the input, it needs $4 \log N$ public key encryptions and $2 \log N$ public key decryptions. Lin et al.[29] propose a two-round protocol to solve the Millionaires' Problem. Their protocol uses multiplicative homomorphic encryption scheme and is more efficient than an additive one practically. It can save computation time and communication bandwidth in practicality. Let p be the modulo prime, the solution in [29] takes $5n \log p$ modular multiplications. And the size of exchanged messages is $6n \log p$ bits in [29]. Fully homomorphic encryption schemes can support more operations over the chiptexts[30], but they are still inefficient and unpractical.

Shundong et al.[31] propose a symmetric cryptographic solution to the millionaires' problem based on set-inclusion problems using a commutative encryption scheme and claims that it is more efficient for practical applications than known solutions and is capable of greatly reducing the computational cost. Unfortunately, we have discovered that the solution has some security flaws and is not more efficient than our protocol based on public key cryptography when the size of the input is large.

Veugen et al.[32] have analyzed the state-of-the-art comparison protocols. In terms of execution time, they point that Damgard's solution[33] developed on the basis of the dedicated DGK homomorphic encryption scheme outperforms the other protocols. Nevertheless, this solution has an initialization time of approximately 154 seconds for medium term security.

Our contributions can be summarized as follows:

1) We find and analyze some security flaws of Shundong's symmetric cryptographic solution to millionaires' problem and this solution is not more efficient and practical than some previous solutions. And the existing problems are formulated and demonstrated, and then their work are also discussed in this paper.

2) We introduce a new solution based on the Decisional Diffie-Hellman assumption as well as the set intersection problem to the millionaires' problem and . Experimental results show that our solution is more efficient and practical.

3) We find that the performance of our solution to the millionaires' problem can be further improved by the random secret keys size setting to be more less bits without having degenerated the underlying intractable discrete logarithm problem. Consequently, our solution is also more efficient than the state-of-the-art solution without pre-computations.

This paper is an improved version of our conference paper[34]. Based on the conference paper[34], we have made significant improvements on our work.

The rest of the paper is organized as follows. Section 2 provides some discussions on Shundong's symmetric cryptographic solution to millionaires' problem. In Section 3 we propose our solution to Yao's millionaires' problem. In Section 4, we demonstrate security analysis and proof to our solution. Section 5 possesses experiment results as well as analyses. Ultimately, the paper will be concluded in Section 6.

2 | SOME DISCUSSIONS ON SHUNDONG'S SYMMETRIC CRYPTOGRAPHIC SOLUTION TO MILLIONAIRES' PROBLEM

Shundong's symmetric cryptographic solution to millionaires' problem is proposed based on a private set-inclusion problem. This problem can be formally defined as follows: Alice has a private number x, and Bob has a private set $X = \{x_1, x_2, x_3, ..., x_n\}$. Alice and Bob need to know whether $x \in X$ without disclosing their private data either x or X to the counterpart. It can be solved with a commutative encryption scheme that has been made for the purpose of determining whether the two numbers are equal[35]. The commutative encryption scheme can be either an asymmetric encryption scheme or a symmetric encryption scheme. In fact, the set-inclusion problem is a special case of the private intersection problem. Agrawal et al.[36] propose a solution with a commutative encryption based on the Decisional Diffie-Hellman assumption to solve the private intersection problem. Neither of the two parties could learn the other party's information outside of the intersection because of lacking necessary key

Protocol 1: Set-inclusion problem[31]

Inputs:
Alice: x
Bob: X

Output: Whether $x \in X$.

The protocol:

1. Preparation:

   Bob chooses a subset A from $\{X, \bar{X}\}$, whose cardinality is smaller than $m/2$. Assume that $A = \{a_1, a_2, ..., a_k\}$. Based on A, Bob defines a new set $B = \{b_1, b_2, ..., b_k, b_{k+1}, ..., b_t\}$, where $t = \lfloor m/2 \rfloor$, $b_i = a_i$ for $1 \leq i \leq k$, whereas $b_i \notin U$ for $k+1 \leq i \leq t$. Obviously, $A \subset B$ and $A \cap B$=A. If $x \in B$, then $x \in A$, and B hide $|A|$, and hides $|X|$.

2. Bob encrypts B using a commutative encryption scheme and obtains $E_{K_b}(B) = \{E_{K_b}(b_1), E_{K_b}(b_2), ..., E_{K_b}(b_t)\}$; Bob sends $E_{K_b}(B)$ to Alice.

3. Alice encrypts $E_{K_b}(B)$ and x with the same commutative encryption scheme and gets $E_{K_a}(E_{K_b}(B))$ and $E_{K_a}(x)$; And she sends $E_{K_a}(E_{K_b}(B))$ and $E_{K_a}(x)$ to Bob.

4. Bob encrypts $E_{K_a}(x)$ and gets $E_{K_b}(E_{K_a}(x))$, determines whether $E_{K_b}(E_{K_a}(x)) \in E_{K_a}(E_{K_b}(B))$. An then determines whether $x \in X$ as follows:

   If $(E_{K_b}(E_{K_a}(x)) \in E_{K_a}(E_{K_b}(B)) \wedge (A = X)$, then $x \in X$, else $x \notin X$.

   If $(E_{K_b}(E_{K_a}(x)) \notin E_{K_a}(E_{K_b}(B)) \wedge (A = \bar{X})$, then $x \in X$, else $x \notin X$.

information. A similar protocol is also proposed by Li et al. based on public key cryptography[37]. However, the two solutions suffer the same questions of more computational complexity and can also reveal $|X|$. Consequently, Shundong et al. introduce a new solution based on symmetric encryption scheme, which is a commutative encryption scheme. And the new solution can be efficient and maintains the privacy of $|X|$[31].

In simple terms, a commutative encryption scheme must satisfy that $E_a(E_b(x)) = E_b(E_a(x))$, where E is an encryption function and a and b are two specified keys. First, the formal protocol with a commutative encryption to the set-inclusion problem is defined as Protocol 1. Let's suppose that U is a complete set, $m = |U|$, $X \subseteq U$ and $x \in X$. The complementary set of X is denoted by $\bar{X}$, and thus $U = X \cup \bar{X}$, $|U| = |X| + |\bar{X}|$.

Through the applications of a commutative scheme, a symmetric encryption solution based on Protocol 1 to the set-inclusion problem proposed in reference [31] is defined as Protocol 2.

Shundong et al. have analyzed the security of Protocol 2 and proved that it is secure in reference[31]. But in fact, Protocol 2 exhibits some important security drawbacks.

First, we discover a definition flaw in Protocol 1. If the cardinality $|U|$ of the set U is even, Bob could not determine the subset A from $\{X, \bar{X}\}$ because the cardinality of the set X could equal to the cardinality of the set $\bar{X}$ according to Protocol 1. A simply solution is that an element $y \notin U$ can be added into the set X. So the cardinality of the new set X' is odd and it will not influence the result. For simplicity, this flaw will not be considered for later discussion.

There exist two important security drawbacks in Protocol 2. The first one is found by Xie et al.[16]. They find that Alice could easily explore Bob's whole set X if Alice has known the set U. For each element $e \in U$, Alice can easily find r'_i such that $e \oplus r'_i = x \oplus r_i$. When Alice receives the two sets of D and $\pi(E) = \{e_{\pi(1)}, e_{\pi(2)}, ..., e_{\pi(t)}\}$ from Bob in step 5, instead of computing $\{b_1 \oplus s_1 \oplus r_1, b_1 \oplus s_2 \oplus r_2, ..., b_t \oplus s_t \oplus r_t\}$, she could compute $G' = D \oplus R' = \{b_1 \oplus s_1 \oplus r'_1, b_1 \oplus s_2 \oplus r'_2, ..., b_t \oplus s_t \oplus r'_t\}$. If $|\pi(E) \cap G'| = 1$, then $e \in B$. So Alice can determine the set X according to the result of protocol 2.

We find the second security drawback. Suppose that $A = X$ and $x \notin X$. If $\exists r_i \in R, s_j \in S, r_m \in R, s_n \in S, b_q \in B$ and $x \neq b_q$ such that $x \oplus r_i \oplus s_j = b_q \oplus r_m \oplus s_n$, then $|\pi(E) \cap G| = 1$. According to protocol 2, if $A = X$ then $x \in X$. But in fact, if $x \neq b_q$ and $x \notin X$, the result can still show $x \in X$. A simple example is shown as follows:

Protocol 2: A symmetric encryption solution to the set-inclusion problem

Inputs:
Alice: x
Bob: X

Output: Whether $x \in X$.

The protocol:

1. Preparation:

   Bob prepares X, $\bar{X}$, A, B according to Protocol 1.

2. Alice generates a random sequence $R = \{r_1, r_2, ..., r_t\}$, and Bob generates another random sequence $S = \{s_1, s_2, ..., s_t\}$.
3. Alice computes $C = x \oplus R = \{x \oplus r_1, x \oplus r_2, ..., x \oplus r_t\} = \{c_1, c_2, ..., c_t\}$, and sends C to Bob.
4. Bob computes

   $D = B \oplus S = \{b_1 \oplus s_1, b_2 \oplus s_2, ..., b_t \oplus s_t\} = \{d_1, d_2, ..., d_t\}$,

   $E = C \oplus S = \{c_1 \oplus s_1, c_2 \oplus s_2, ..., c_t \oplus s_t\}$=$\{x \oplus r_1 \oplus s_1, x \oplus r_2 \oplus s_2, ..., x \oplus r_t \oplus s_t\} = \{e_1, e_2, ..., e_t\}$.
5. Bob generates a random permutation of E, expressed by $\pi(E)$, and sends D and $\pi(E) = \{e_{\pi(1)}, e_{\pi(2)}, ..., e_{\pi(t)}\}$ to Alice.
6. Alice computes $G = D \oplus R = \{b_1 \oplus s_1 \oplus r_1, b_2 \oplus s_2 \oplus r_2, ..., b_t \oplus s_t \oplus r_t\}$ and $|\pi(E) \cap G|$. Alice sends the result of $|\pi(E) \cap G|$ to Bob.
7. If $|\pi(E) \cap G| = 1$, then $x \in B$, else $x \notin B$. Bob determines whether $x \in X$ as follows:

   If $(x \in B) \wedge (A = X)$, then $x \in X$, else $x \notin X$.

   If $(x \notin B) \wedge (A = \bar{X})$, then $x \in X$, else $x \notin X$.
8. Bob tells Alice the result.

Suppose that $U = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$, $x = 3$ and $X = \{2, 4\}$ and Alice generates a random sequence $R = \{1, 2\}$ and Bob generates a random sequence $S = \{3, 1\}$. Alice can get $D = \{2 \oplus 1 \oplus 3, 4 \oplus 2 \oplus 1\}$ and $\pi(E) = \{3 \oplus 1 \oplus 3, 3 \oplus 2 \oplus 1\}$. We can find that $|\pi(E) \cap G| = |\{1 \oplus 2 \oplus 3\}| = 1$. The result show that $x \in X$, but $x \notin X$. The second security drawback is that the result of protocol 2 could be wrong.

Since Shundong's symmetric encryption solution to the set-inclusion problem has some drawbacks, their symmetric solution to millionaires' problem is also broken. However, even if we postulate that symmetric encryption solution to the set-inclusion problem is perfect (For example, we can construct an asymmetric encryption solution to the set-inclusion problem to instead of protocol 2 to eliminate the drawbacks of Shundong's symmetric cryptographic solution to Yao's millionaires' problem), their symmetric solution to millionaires' problem also exhibits some important security drawbacks.

Let's suppose that a symmetric encryption solution to the set-inclusion problem is secure. In some cases, Alice could easily explore Bob's y in this scheme.

Proposition 1. Suppose that $G = E_{K_a}(E_{K_b}(B)) = \{g_1, g_2, ..., g_t\}$, $x = \lfloor m/2 \rfloor$,if and only if Alice knows $E_{K_b}(E_{K_a}(x)) = g_t$ and $x < y$, then $y = x + 1$.

Proof. For the "if" part:

According protocol 1, if $E_{K_b}(E_{K_a}(x)) = g_t$ then $E_{K_b}(E_{K_a}(x)) \in E_{K_a}(E_{K_b}(B))$. According to protocol 3, if $x < y$ then $x \in X$. According to protocol 1, if $(E_{K_b}(E_{K_a}(x)) \in E_{K_a}(E_{K_b}(B)) \wedge x \in X$, then $A = X$. $E_{K_b}(E_{K_a}(x)) = g_t$, so $b_t = x$. According to protocol 3, $B = A = X = \{1, 2, ..., y - 1\}$, so $b_t = y - 1 = x$, i.e. $y = x + 1$.

For the "only if" part:

Protocol 3: Shundong's Symmetric cryptographic solution to Yao's millionaires' problem

Inputs:
Alice: x
Bob: y

Output:whether $x < y$.
For simplicity, suppose that $0 < x, y < 100$. Let $U = \{1, 2, ..., 99\}$; then $x, y \in U$. Bob constructs a set X as follows: $X = \{1, 2, ..., y - 1\}$. Clearly, $X \in U$. If $x \in X$, then $x < y$, and otherwise $x \geq y$.
The protocol:

1. Preparation:

   Provided x and X, the preparation of X, $\bar{X}$, A, B according to Protocol 1. The protocol proceeds as follows:

2. Alice and Bob use Protocol 2 to determine whether $x \in X$, and Bob obtains the result. Bob concludes that $x < y$ if $x \in X$, and $x \geq y$ otherwise.

3. Bob tells Alice the result.

According to protocol 1 and protocol 3, if $x = \lfloor m/2 \rfloor$ and $y = x+1$, then $X = \{1, 2, ..., y-1\} = \{1, 2, ..., x\}$. And if $x = \lfloor m/2 \rfloor$, then $B = A = X = \{1, 2, ..., x\}$ and $g_t = E_{K_b}(E_{K_a}(b_t)) = E_{K_b}(E_{K_a}(x))$, where $(E_{K_b}(E_{K_a}(x)) \in E_{K_a}(E_{K_b}(B)) \wedge A = X$, so $x \in X$ and according to protocol 3 , $x < y$. So Alice can know that $E_{K_b}(E_{K_a}(x)) = g_t$ and $x < y$. □

Proposition 2. Suppose that $G = E_{K_a}(E_{K_b}(B)) = \{g_1, g_2, ..., g_t\}$ and $x > \lfloor m/2 \rfloor + 2$, if and only if Alice knows $E_{K_b}(E_{K_a}(x)) = g_i$ and $x \geq y$, then $y = x - i + 1$ and $y > \lfloor m/2 \rfloor + 1$.

Proof. For the "if" part:

According protocol 1, if $E_{K_b}(E_{K_a}(x)) = g_i$ then $E_{K_b}(E_{K_a}(x)) \in E_{K_a}(E_{K_b}(B))$. According to protocol 3, if $x \geq y$ then $x \notin X$. According to protocol 1, If $(E_{K_b}(E_{K_a}(x)) \in E_{K_a}(E_{K_b}(B)) \wedge x \notin X$, then $A = \bar{X}$. According to protocol 3, $X = \{1, 2, ..., y - 1\}$, so $y > \lfloor m/2 \rfloor + 1$ and $B = \{b_1, b_2, ..., b_t\} = \{y, y + 1, ..., b_t\}$. If $E_{K_b}(E_{K_a}(x)) = g_i$, then $x = b_i = y + i - 1$, i.e. $y = x - i + 1$.

For the "only if" part:

If $y > \lfloor m/2 \rfloor + 1$ then $X = \{1, 2, ..., y - 1\}$ and $A = \bar{X}$. So $B = \{b_1, b_2, ..., b_t\} = \{y, y + 1, ..., b_t\}$. If $y = x - i + 1$, then $B = \{x-i+1, x-i+2, ..., x, ..., b_t\}$. $E_{K_b}(E_{K_a}(B)) = \{E_{K_b}(E_{K_a}(x-i+1)), E_{K_b}(E_{K_a}(x-i+2)), ..., E_{K_b}(E_{K_a}(x)), ..., E_{K_b}(E_{K_a}(b_t))\}$, so $E_{K_b}(E_{K_a}(x)) = g_i$, where $(E_{K_b}(E_{K_a}(x)) \in E_{K_a}(E_{K_b}(B)) \wedge A = \bar{X}$, so $x \notin X$ and according to protocol 3, $x \geq y$. So Alice can know that $E_{K_b}(E_{K_a}(x)) = g_i$ and $x \geq y$. □

In fact, we only need the sufficient condition of the two propositions. Alice can conclude Bob's y if the sufficient condition of the two propositions is satisfied. So two corollaries can be presented as follows.

Corollary 1. Suppose that $G = E_{K_a}(E_{K_b}(B)) = \{g_1, g_2, ..., g_t\}$, $x = \lfloor m/2 \rfloor$,if Alice knows $E_{K_b}(E_{K_a}(x)) = g_t$ and $x < y$, then Alice can conclude that $y = x + 1$.

Corollary 2. Suppose that $G = E_{K_a}(E_{K_b}(B)) = \{g_1, g_2, ..., g_t\}$ and $x > \lfloor m/2 \rfloor + 2$, if Alice knows $E_{K_b}(E_{K_a}(x)) = g_i$ and $x \geq y$, then Alice can conclude that $y = x - i + 1$.

A simple solution to the problem on Shundong's symmetric cryptographic solution to Yao's millionaires' problem is that Bob generates a random permutation of $X = \{1, 2, ..., y - 1\}$ as the new X. However, as previously mentioned, it is still not secure because of some security drawbacks.

3 | OUR SOLUTION TO YAO'S MILLIONAIRES' PROBLEM

Some previous work based on homomorphic encryption have studied and proposed some efficient protocols to Yao's millionaires' problem. Blake et al.[38] use the additive homomorphic Paillier cryptosystem to construct a two-round protocol to Yao's millionaires' problem. The computation cost is $O(n \log N)$ and the communication cost is $O(n \log N)$. Lin et al.[29] also propose a two-round protocol for solving the Millionaires' Problem using the multiplicative homomorphic encryptions and a special coding for the private inputs. Since multiplicative homomorphic encryption scheme is more efficient than an additive one practically, their solution saves computation time and communication bandwidth in practicality. The ElGamal encryption scheme is a multiplicative homomorphic encryption scheme with the scalaring property. And the Paillier encryption scheme is an additive homomorphic encryption scheme. For efficiency of computation, they modify the scheme so that each decryption takes 1 modular exponentiation without affecting the security of the scheme. Shundong et al. propose to use the XOR operation as the symmetric commutative function and the solution can sharply reduce the computational overhead. Unfortunately, it does have some security flaws. Generally, we have two policies to reduce the computational overhead. The first one is to employ a symmetric encryption scheme, and the second one is to employ an asymmetric encryption scheme but we can greatly reduce the computational number of modular multiplications.

3.1 | 0-encoding and 1-encoding

The main idea reducing the computational number of modular multiplications is to reduce the scale of the set intersection problem. Lin et al.[29] use two special encodings, 0-encoding and 1-encoding.

Let $x = x_n x_{n-1} ... x_1 \in \{0,1\}^n$ be a binary string of length n. The 0-encoding of x is the set S_x^0 of binary string x such that

$$S_x^0 = \{x_n x_{n-1} ... x_{i+1} 1 | x_i = 0, 1 \le i \le n\}$$

The 1-encoding of x is the set S_x^1 of binary string such that

$$S_x^1 = \{x_n x_{n-1} ... x_i | x_i = 1, 1 \le i \le n\}$$

Both S_x^1 and S_x^0 have at most n elements.

We can encode x into its 1-encoding S_x^1 and y into its 0-encoding S_y^0.

Theorem 1. x is greater than y if and only if S_x^1 and S_y^0 have a common element.

The proof of theorem 1 and more information about 0-encoding and 1-encoding can be found in [29].

We give an example. Let $x = 10 = 1010_2$ and $y = 6 = 0110_2$ of length 4 (we fill in the leading zeros). We have $S_x^1 = \{1, 101\}$ and $S_y^0 = \{1, 0111\}$. Since $S_x^1 \cap S_y^0 \neq \emptyset$, we have $x > y$. And if $x = 6 = 0110_2$ and $y = 10 = 1010_2$,we have $S_x^1 = \{01, 011\}$ and $S_y^0 = \{11, 1011\}$.Since $S_x^1 \cap S_y^0 = \emptyset$, we have $x \le y$.

In order to construct our solution, we redefine a new 0-encoding and 1-encoding.

Definition 1. Let $x = x_n x_{n-1} ... x_1 \in \{0,1\}^n$ be a binary string of length n. The 0-encoding of x is the set S_x^0 of binary numbers x such that

$$S_x^0 = \{x_n x_{n-1} ... x_{i+1} 1 \underbrace{0_{i-1}, ..., 0_2, 0_1}_{i-1} | x_i = 0, 1 \le i \le n\}$$

The 1-encoding of x is the set S_x^1 of binary numbers such that

$$S_x^1 = \{x_n x_{n-1} ... x_i \underbrace{0_{i-1}, ..., 0_2, 0_1}_{i-1} | x_i = 1, 1 \le i \le n\}$$

Both S_x^1 and S_x^0 have at most n elements.

We also give an example. Let $x = 10 = 1010_2$ and $y = 6 = 0110_2$ of length 4. We have $S_x^1 = \{1000_2, 1010_2\} = \{8, 10\}$ and $S_y^0 = \{1000_2, 0111_2\} = \{8, 7\}$. Since $S_x^1 \cap S_y^0 \neq \emptyset$, we have $x > y$. And if $x = 6 = 0110_2$ and $y = 10 = 1010_2$,we have $S_x^1 = \{0100_2, 0110_2\} = \{4, 6\}$ and $S_y^0 = \{1100_2, 1011_2\} = \{1100_2, 1011_2\} = \{12, 11\}$. Since $S_x^1 \cap S_y^0 = \emptyset$, we have $x \le y$.

3.2 | Our commutative encryption scheme

We propose a commutative encryption scheme solution to Yao's millionaires' problem. The commutative encryption scheme is constructed based on the Decisional Diffie-Hellman assumption. Our commutative encryption scheme requires 1 modular exponentiation for each party, so it is more efficient than multiplicative homomorphic encryption scheme.

Definition 2. Let M denote a message space and K denote a key space. A commutative encryption function is a computable (in polynomial time) and bijection function $f : M \times K \to M$ that satisfies that we have $f_b \circ f_a(m)$=$f_a \circ f_b(m)$, for a given $m \in M$, any $a, b \in K$.

Fact 1. Let M be the group of quadratic residues modulo a prime p, where p is a large 'safe' prime number, i.e., both p and $q = (p-1)/2$ are large primes. Let K be $\{1, 2, ..., q-1\}$. According to Decisional Diffie-Hellman assumption, the power function

$$f_e(m) \equiv m^e \bmod p$$

is commutative encryption function.

(1) $f_b \circ f_a(m) = (m^a \bmod p)^b \bmod p$=$m^{ab} \bmod p = (m^b \bmod p)^a \bmod p = f_a \circ f_b(m)$

(2) Each of the powers f_e is a bijection.

The DDH is a computational hardness assumption about a certain problem of discrete logarithms in cyclic groups. So the security of f_e depends on the computational difficulty of discrete logarithm problem. Let h denote a public collision-free hash function.

Our solution (Protocol 4) is constructed based on the Decisional Diffie-Hellman assumption. Let p be the quadratic residues modulo prime. Let n be the length of the private inputs of x and y. The most time-consuming computation is modular multiplications, so we will only count the cost of modular multiplications. Our solution takes no more than $4n \log \frac{p}{2}$ modular multiplications and the solution in[29] takes $5n \log p$ modular multiplications. The size of exchanged messages in our solution is no more than $3n \log p$ bits and it is $6n \log p$ bits in[29].

4 | SECURITY ANALYSIS

Goldreich[39] presents a security evaluation benchmark based on the simulation paradigm, which has been widely used to prove the secure of a multiparty computation solution.

4.1 | The Semi-Honest Model

We suppose that both of the parties in our solution to Yao's millionaires' problem are semi-honest. Loosely speaking, a semi-honest party is one who follows each step of the protocol properly, except that it collects all its intermediate computation results and might attempt to infer the other parties' private inputs from the final and intermediate computation results. Roughly speaking, a protocol is private in the semi-honest model if each party is unable to conclude the private input data of another party from the final and his/her collected intermediate computation results. And our solution is privacy preserving in a semi-honest setting.

4.2 | Formulation of Privacy

Goldreich[39] proposes the privacy definition of secure multiparty computation to study the security of multiparty computation schemes. Let $f = (f_1, f_2)$ be a probabilistic polynomial-time functionality and Π be a two-party protocol for computing f. The *view* of the first party during an execution of Π on the input (x, y), denoted by $\text{view}_1^{\Pi}(x, y)$, is $(x, r^1, m_1^1, ..., m_t^1)$, where r^1 represents the outcome of the first party's internal coin tosses, and m_i^1 represents the i-th message it has received. The output of the first party during an execution of Π on the input (x, y), denoted by $\text{output}_1^{\Pi}(x, y)$, is implicit in the party's view of the execution. The view and output of the second party can be defined analogously.

Definition 3. For a functionality f, Π privately computes f if there exist probabilistic polynomial-time algorithms, denoted by S_1 and S_2 such that

$$\begin{gathered}\{(S_1(x, f_1(x, y)), f_2(x, y))\}_{x,y} \overset{c}{\equiv} \\ \{(\text{view}_1^{\Pi}(x, y), \text{output}_2^{\Pi}(x, y))\}_{x,y},\end{gathered} \quad (1)$$

Protocol 4: Our solution to Yao's millionaires' problem

Inputs:
Alice: x and a large safe prime p, where $0 < x < 2^n, n = \lfloor \log(p-1)/2 \rfloor$
Bob: y and a large safe prime p, where $0 < y < 2^n, n = \lfloor \log(p-1)/2 \rfloor$

Output: whether $x > y$.

The protocol:

1. Alice:
   (a) Alice generates a random secret key a, where a is a large number and $a < (p-1)/2$.
   (b) Alice computes S_x^1.
   (c) for each $s \in S_x^1$ computes $S = S \cup \{f_a(h(s)^2)\}$.
   (d) Alice prepares $l_1 = n - |S_x^1|$ random numbers z_j and combines them to the set S, where $z_j \in M$, $1 \leq j \leq l_1$ and each z_j is unique.
   (e) Alice generates a random permutation of S, expressed by $\pi_1(S)$ and sends $\pi_1(S)$ to Bob.
2. Bob:
   (a) Bob generates a random secret key b, where b is a large number and $b < (p-1)/2$, then Bob gets $\pi_1(S)$ and computes $G = f_b(\pi_1(S))$.
   (b) Bob computes S_y^0.
   (c) for each $r \in S_y^0$ computes $R = R \cup f_b(h(r)^2)$.
   (d) Bob prepares $l_2 = n - |S_y^0|$ random numbers z_j and combines them to the set R, where $z_j \in M$, $1 \leq j \leq l_1$ and each z_j is unique.
   (e) Bob generates a random permutation of R, expressed by $\pi_2(R)$ and sends $\pi_2(R)$ to Alice.
   (f) Bob generates a random permutation of G, expressed by $\pi_3(G)$ and sends $\pi_3(G)$ to Alice.
3. Alice gets $\pi_2(R)$ and computes $H = f_a(\pi_2(R))$.
   If $|H \cap \pi_3(G)| = 1$, Alice concludes that $x > y$ and $x \leq y$ otherwise. Alice tells Bob the result.

and

$$\begin{aligned} &\{(S_2(y, f_2(x,y)), f_1(x,y))\}_{x,y} \overset{c}{\equiv} \\ &\{(\mathrm{view}_2^{\Pi}(x,y), \mathrm{output}_1^{\Pi}(x,y))\}_{x,y}, \end{aligned} \tag{2}$$

where $\overset{c}{\equiv}$ denotes computational indistinguishability, $\mathrm{view}_1^{\Pi}(x,y)$ and $\mathrm{view}_2^{\Pi}(x,y)$, $\mathrm{output}_1^{\Pi}(x,y)$ and $\mathrm{output}_2^{\Pi}(x,y)$, are related random variables, defined as a function of the same random execution.

4.3 | Security Analysis on Our Solution

Notice that in this protocol $f_1(x,y) = f_2(x,y) = x > y$ or $f_1(x,y) = f_2(x,y) = x \leq y$, and the *view* of a party is defined by $(x, r, m_1, m_2, ...)$, where x is the party's input, r is the private coin tosses, and m_i is the i-th message it received.

Suppose that $f_1(x,y) = f_2(x,y) = x > y$. We can construct simulator S_1 as follows:

S_1 receives $(x, f_1(x,y))$ as its input, and simulates $\mathrm{view}_1^{\Pi}(x,y)$ is satisfied by eq. 1.

1. S_1 first generates a random secret key b', where b' is a large number and $b' < (p-1)/2$. And then S_1 randomly constructs a number y' such that $|S_x^1 \cap S_{y'}^0| = 1$.

2. S_1 prepares $l_2' = n - |S_{y'}^0|$ random numbers z_j.

3. According to protocol 4, S_1 computes S_x^1, $S_{y'}^0$, S and R'.
4. S_1 computes $\pi_1(S)$, and $\pi_2(R')$.
5. S_1 computes G', H' and $\pi_3(G')$.

Let $S_1(x, x > y) = \{x, a, b', S_x^1, S_{y'}^0, S, R', \pi_1(S), \pi_2(R'), G', \pi_3(G'), H', |H' \cap \pi_3(G')| = 1\}$. Since $\text{view}_1^{\Pi}(x, y) = \{x, a, S_x^1, S, \pi_1(S), \pi_2(R'), \pi_3(G'), H', |H' \cap \pi_3(G')| = 1\}$. So it shows that

$\{(S_1(x, f_1(x, y)), f_2(x, y))\}_{x,y} \overset{c}{\equiv} \{(\text{view}_1^{\Pi}(x, y), \text{output}_2^{\Pi}(x, y))\}_{x,y}$.

Simulator S_2 for simulating $\text{view}_2^{\Pi}(x, y)$, such that

$\{(S_2(x, f_2(x, y)), f_1(x, y))\}_{x,y} \overset{c}{\equiv} \{(\text{view}_2^{\Pi}(x, y), \text{output}_1^{\Pi}(x, y))\}_{x,y}$.

Similarly, if $f_1(x, y) = f_2(x, y) = x \leq y$, we can construct two simulators S_1 and S_2, such that

$\{(S_1(x, f_1(x, y)), f_2(x, y))\}_{x,y} \overset{c}{\equiv} \{(\text{view}_1^{\Pi}(x, y), \text{output}_2^{\Pi}(x, y))\}_{x,y}$,

and

$\{(S_2(x, f_2(x, y)), f_1(x, y))\}_{x,y} \overset{c}{\equiv} \{(\text{view}_2^{\Pi}(x, y), \text{output}_1^{\Pi}(x, y))\}_{x,y}$.

5 | EXPERIMENTAL RESULTS AND ANALYSIS

5.1 | Experiment Settings

In order to demonstrate the fact efficiency of our solution, we implemented our protocol using Python 2.7 based on Charm-Crypto[40] which depends on a few open-source C math libraries including OpenSSL, GMP (GNU Multiple Precision Arithmetic Library) and PBC (Pairing-based Cryptography Library). We builded the Charm-Crypto based on GMP 6.0.0[41] not using the side-channel silent mpz_powm_sec function. And all of the experiments have been carried out on a machine running the Ubuntu subsystem in Windows 10 System with an Intel i5-4690 Processor at 3.50GHz and 8GB RAM. The asymmetric cryptographic key lengths have been chosen according to the current NIST standard from 1024 to 8192 bits.

5.2 | Comparison with the Solution based on Symmetric Commutative Encryption Scheme

Because the efficiency of public key encryption schemes appears less than 0.1% of symmetric encryption schemes, no solution developed on the bases of the public key cryptography to Yao's millionaires' problem can be efficient[31]. Let N indicate the maximal value of the input in protocol 2 and its bit size is $n = \lceil \log N \rceil$. Protocol 2 takes $4N$ XOR operation. When N is a small number, It is inevitable that protocol 2 is more efficient than our protocol. For the demonstration of the fact computation cost of the solution based on symmetric commutative encryption scheme and asymmetric commutative encryption scheme, we implemented our protocol 4 and XOR operation in protocol 2 in accordance with the previous experiment settings.

We test the performance of XOR in Protocl 2 and results have been presented in Fig. 1 and Fig. 2. The input size is chosen from 8 to 32 bits. As evident from the Fig. 1 and Fig. 2, it can be clearly observed that, with the linear increase in the size of an input, the processing time of Protocol 2 is increased linearly.

The Diffie-Hellman "group" is used for public cryptographic schemes. These groups are approximately as strong as a symmetric key. We choose 1024 (Group 2), 1536(Group 5), 2048(Group 14), 3072(Group 15), 4096(Group 16), 6144(Group 17) and 8192(Group 18) bits Diffie-Hellman groups to test the performance of our Protocol 4. The results of processing time are summarized in Table 1.

As suggested by Table 1 and Fig. 3, we can observe that as the size of an input increases linearly the cost is almost increased linearly for the same modulus size. Nevertheless, from Table 1, Fig. 4 and Fig. 5, we can observe that as the modulus size increases linearly the cost is not increased linearly for the same input size.

The security level of our solution is determined by the modulus size, and it is more secure with the longer modulus size and the cost also is more expensive. Accordingly, we can determine the modulus size in accordance with security strength. From Table 1, without considering the intersection operation, we can observe the fact that the computation cost of the XOR operation in protocol 2 is very small if the size of the private input is also small, otherwise the cost is large, for instance when the input size bit is 32. When the input size bit is 16, the cost for the computation of XOR operation is 5.419 ms. Thus, we can conclude the cost to be no less than 355,139.584 ($5.419 * 2^{16}$) ms and it is actually 414,472.559 ms. And we can continue to conclude the cost to be no less than 56,448 years when the input size bit is 64. So protocol 2 is too expensive to be practical when the input size

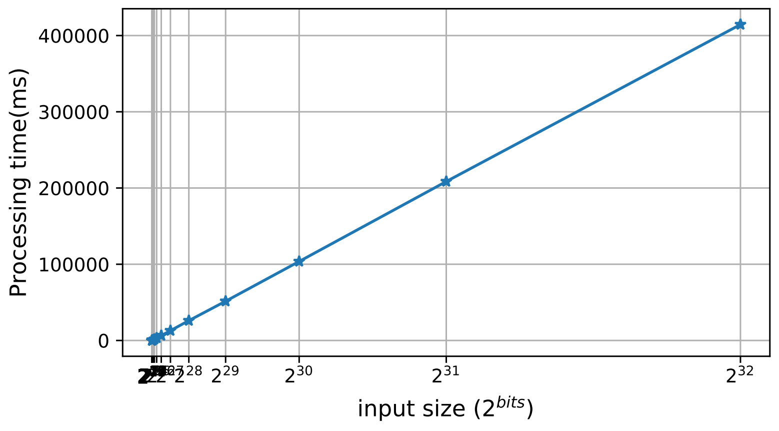


FIGURE 1 Processing time of Protocol 2 vs. input size(2^{bits})

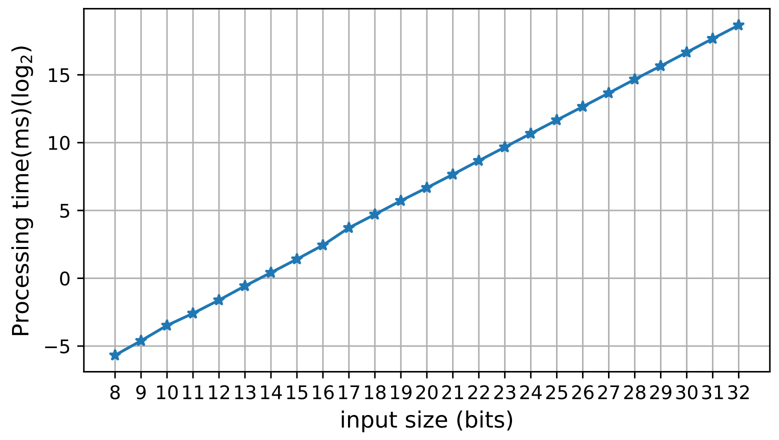


FIGURE 2 Processing time($log_2(y)$) of Protocol 2 vs. input size(bits)

TABLE 1 Comparison of processing time (ms) between our and Shundong's Protocol

Input size (bits)	Processing time(ms) of Protocol 4 for 7 kinds of modulus size(bits)							XOR in Protocol 2
	1024	1536	2048	3072	4096	6144	8192	
8	14	39	85	265	601	1680	3560	0.02
16	26	78	171	529	1199	3360	7106	5.419
32	52	154	341	1056	2403	6724	14,361	414,472.559
64	104	308	681	2112	4805	13,441	28,674	-
128	209	617	1363	4223	9614	26,871	571,77	-
256	432	1270	2734	8462	19,215	53,785	114,074	-
512	891	2520	5512	16,977	38,510	107,623	228,183	-

is large. Thus, our solution to Yao's millionaires' problem is more efficient and practical than the solution based on symmetric commutative encryption scheme.

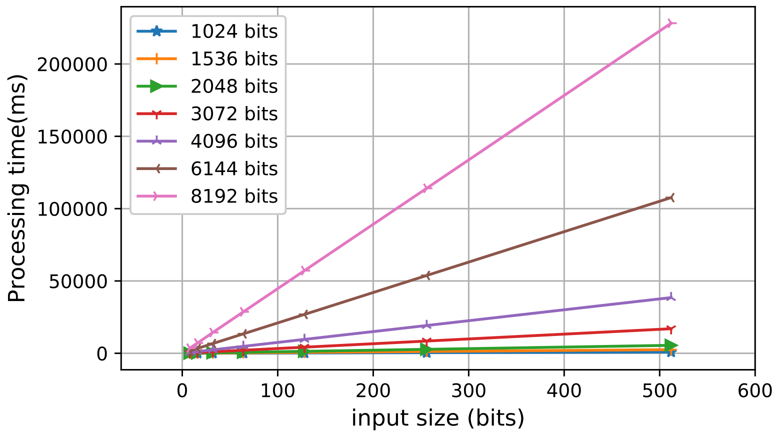


FIGURE 3 Processing time of Protocol 4 vs. input size(bits)

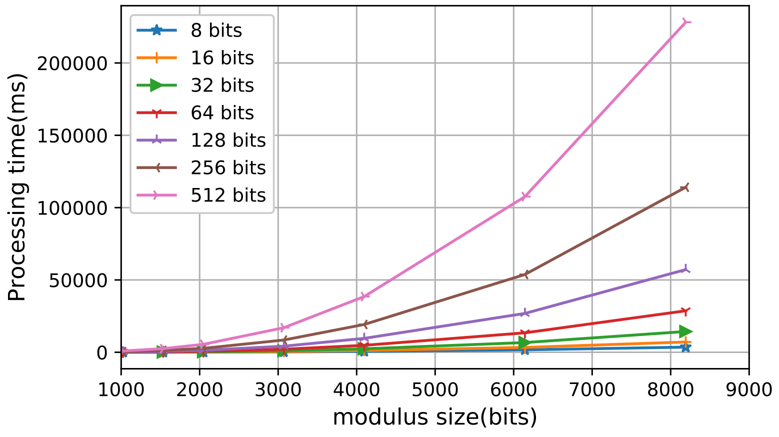


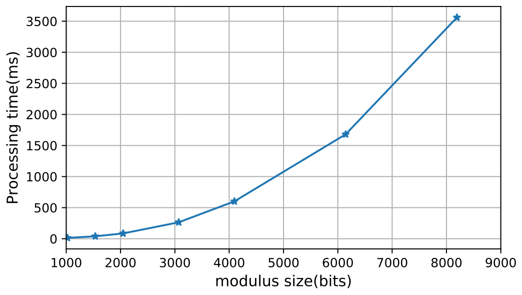
FIGURE 4 Processing time of Protocol 4 vs. modulus size(bits)

5.3 | Efficiency Improvement

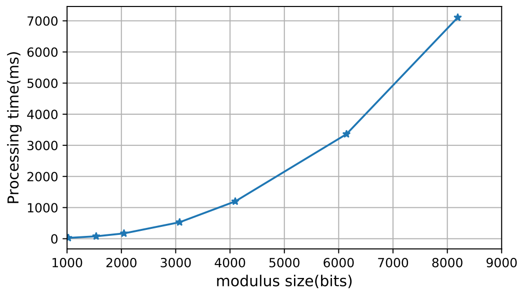
It should be taken into notice that the parameter p is a large safe prime chosen in Protocol 4, i.e., both p and $q = (p - 1)/2$ are large primes. In fact, the parameter p could be such a prime that $p - 1$ has a sufficiently large prime factor q. "sufficiently large" means that the size of q is at least 160 bits, i.e., $q > 2^{160}$. For short-term security, the 2^{160} setting is imposed by the lower-bound requirement of the index computation attack algorithm called λ-method or kangaroo method for solving the discrete logarithm problem proposed by Pollard[42]. Thus, the random secret keys size setting for a and b in Protocol 4 can be 160 bits without having degenerated the underlying intractable discrete logarithm problem[43]. The performance of our solution to Yao's millionaires' problem can be further improved. The improved results are summarized in Table 2, Fig. 6, Fig. 7 and Fig. 8 while $a \approx 2^{160}$ and $b \approx 2^{160}$. It should be taken into notice that the processing time of Protocol 4 for $a \approx 2^{160}$ and $b \approx 2^{160}$ may have been reduced by 4 times while the parameter size of p is 1024 bits.

5.4 | Comparison with the State-of-the-art Solution

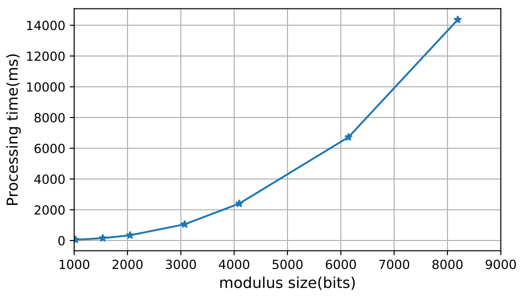
In terms of processing time, Damgard's solution[33] based on the dedicated DGK homomorphic encryption scheme is the state-of-the-art comparison protocol. But it has to spend around 154 seconds during its initialization phase for a 24-bit input and medium-term security reported in[32]. The DGK homomorphic encryption scheme has been found to be not secure, and Damgard



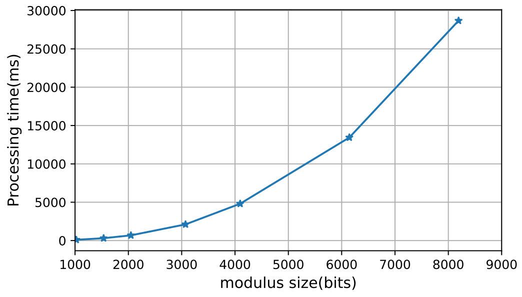
(a) 8 bits



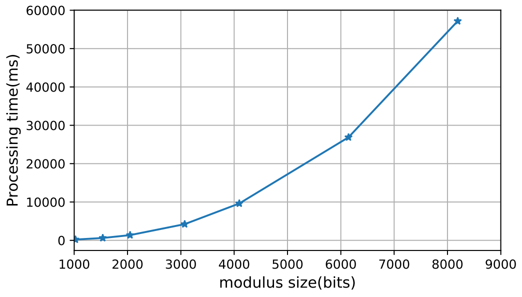
(b) 16 bits



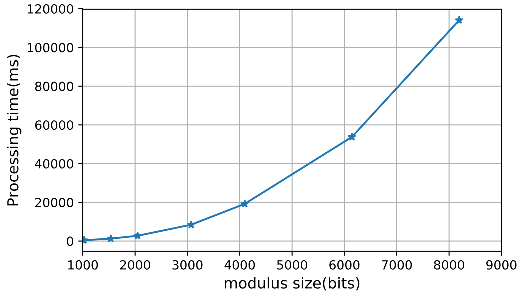
(c) 32 bits



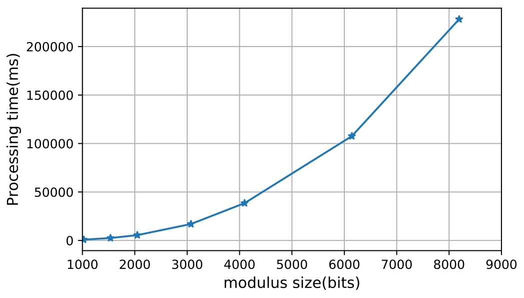
(d) 64 bits



(e) 128 bits



(f) 256 bits



(g) 512 bits

FIGURE 5 Processing time of Protocol 4 vs. modulus size(bits)

et al. have described a correction to the cryptosystem[44]. The correction involves much more initialization time. Owed to the fact that DGK encryption function requires one complex modular exponentiation, it can be computed in advance during idle times. By this optimization, one encryption of DGK homomorphic encryption scheme requires only one expensive modular exponentiation and one modular multiplication.

TABLE 2 Improved experiment results for $a \approx 2^{160}$ and $b \approx 2^{160}$

Input size (bits)	Processing time(ms) of Protocol 4 for 7 kinds of modulus size(bits)						
	1024	1536	2048	3072	4096	6144	8192
8	3	6	9	17	29	53	84
16	6	10	17	34	56	105	168
32	11	21	34	67	113	210	333
64	22	40	66	133	222	421	665
128	46	83	134	268	450	840	1333
256	99	177	275	544	909	1685	2689
512	237	392	594	1136	1871	3446	5466

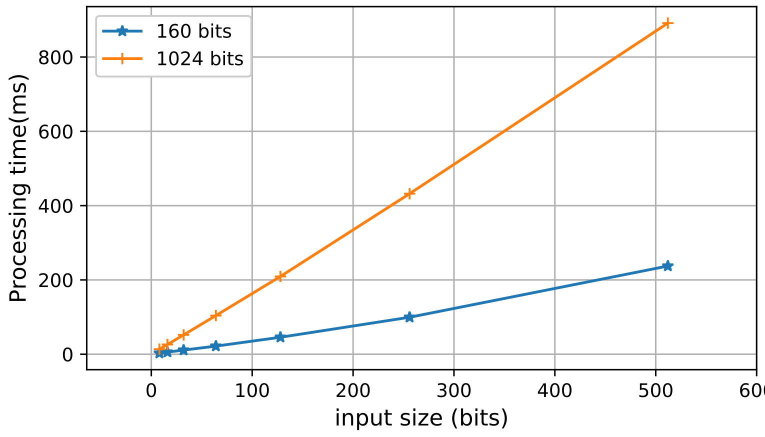


FIGURE 6 Comparison of Processing time of Protocol 4 for the key size is 160 bits and 1024 bits

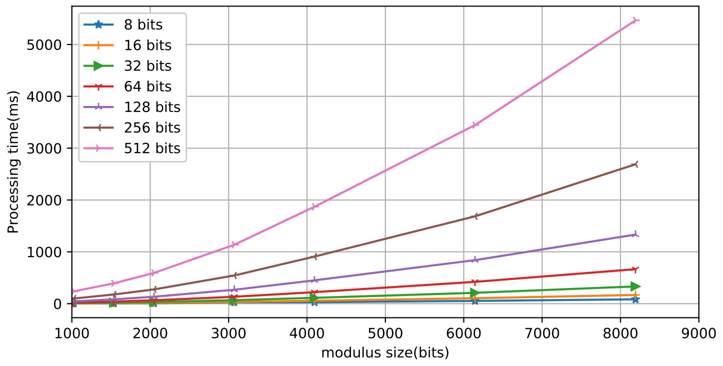
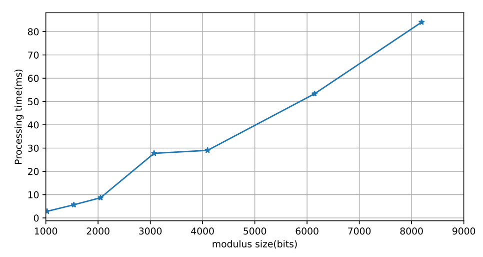
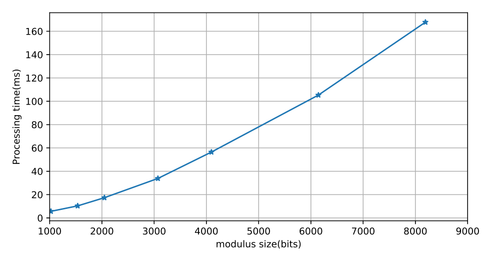


FIGURE 7 Processing time of Protocol 4 for $a \approx 2^{160}$ and $b \approx 2^{160}$ vs. modulus size(bits)

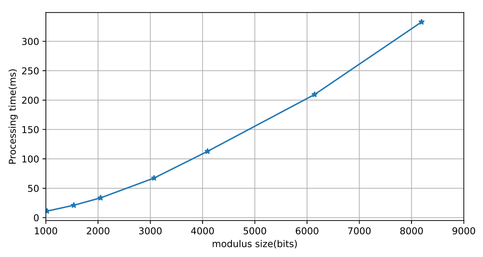
We first give some notations to represent the computation cost of basic computational operation in the algorithms of DGK cryptosystem in Table 3. Accordingly, computation cost will be counted in the number of modular multiplication.



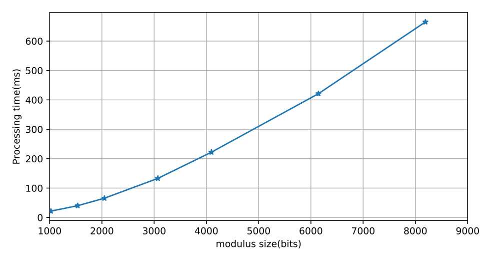
(a) 8 bits



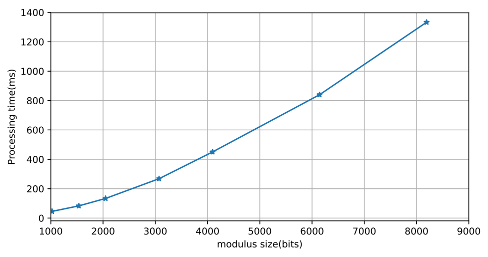
(b) 16 bits



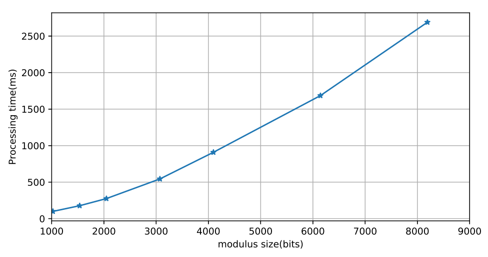
(c) 32 bits



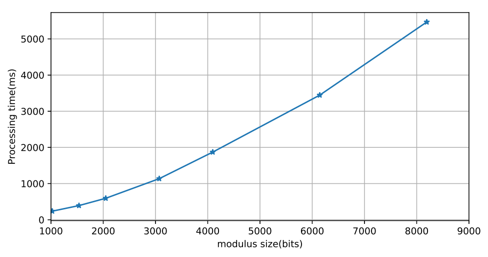
(d) 64 bits



(e) 128 bits



(f) 256 bits



(g) 512 bits

FIGURE 8 Processing time of Protocol 4 for $a \approx 2^{160}$ and $b \approx 2^{160}$ vs. modulus size(bits)

By analysis, we can obtain the processing cost of Damgard's solution[33] without pre-computation. Alice's computation cost can be expressed as follows:

$$T_A = nt_d + nt_e \tag{3}$$

And Bob's computation cost can be expressed as follows:

$$\begin{aligned} T_B &= t_e + nt_e + t_e + 2t_a + t_i + (n-1)\left(3t_a + t_i + 2t_a + 3t_a + t_i + t_{p=l} + t_{p=2t} + t_a\right) \\ &= (n+2)\,t_e + 2t_a + t_i + (n-1)\left(9t_a + 2t_i + t_{p=2t+l}\right) \\ &= (n+2)\,t_e + (9n-7)\,t_a + (2n+1)\,t_i + (n-1)\,t_{p=2t+l} \end{aligned} \tag{4}$$

TABLE 3 The computation cost of basic computational operations for DGK

Symbol	computational operation	without pre-computation	with pre-computation
t_e	Encryption	$l + 2t + 1$	$l + 1$
t_d	Decryption	$t/2$	$t/2$
t_a	Addition of two ciphertexts	2	2
$t_{p=x}$	Exponentiation of a ciphertext	x	x
t_i	Invert	l	l

1. l represents the bit length of prime u.
2. t represents the bit length of prime v_p and v_q.
3. x represents the bit length of the exponential value.

According to the computation cost of basic computational operations for DGK in Table 3, we can obtain the processing cost of Damgard's solution according to the number of modular multiplication without pre-computations.

$$\begin{aligned} T_A &= nt_d + nt_e \\ &= nt/2 + n(l + 2t + 1) \end{aligned} \tag{5}$$

$$\begin{aligned} T_B &= (n+2)\,t_e + (9n-7)\,t_a + (2n+1)\,t_i + (n-1)\,t_{p=2t+l} \\ &= (n+2)(l+2t+1) + 2(9n-7) + (2n+1)\,l + (n-1)(2t+l) \\ &= 4nt + 4nl + 2l + 2t + 19n - 12 \end{aligned} \tag{6}$$

$$\begin{aligned} T_{Total} &= T_A + T_B \\ &= nt/2 + n(l+2t+1) + 4nt + 4nl + 2l + 2t + 19n - 12 \\ &= 6.5nt + 5nl + 2l + 2t + 20n - 12 \end{aligned} \tag{7}$$

we can also obtain the processing cost of Damgard's solution according to the number of modular multiplication with pre-computation.

$$\begin{aligned} T'_A &= nt_d + nt_e \\ &= nt/2 + n(l+1) \end{aligned} \tag{8}$$

$$\begin{aligned} T'_B &= (n+2)\,t_e + (9n-7)\,t_a + (2n+1)\,t_i + (n-1)\,t_{p=l} \\ &= (n+2)(l+1) + 2(9n-7) + (2n+1)\,l + (n-1)\,l \\ &= 4nl + 2l + 19n - 12 \end{aligned} \tag{9}$$

$$\begin{aligned} T'_{Total} &= T'_A + T'_B \\ &= nt/2 + n(l+1) + 4nl + 2l + 19n - 12 \\ &= 0.5nt + 5nl + 2l + 20n - 12 \end{aligned} \tag{10}$$

The comparison results of computational with several solutions are summarized in Table 4. S and G in our Protocol 4 can also be computed in advance during idle times. From Table 4, we can see that our solution either with or without pre-computations outperforms Lin et al.'s solutions according to the number of modular multiplication. In addition, our solution without pre-computations also outperforms the state-of-the-art comparison protocol based on DGK homomorphic encryption scheme. However, while $t < l$, our solution with pre-computations cannot outperform the state-of-the-art comparison protocol. Thus, our solution requires the lest computation cost if the complex modular exponentiation cannot be computed in advance during idle times. If the bit length of private inputs is large enough for Alice and Bob, we solution can also compare well with the state-of-the-art comparison protocol while the complex modular exponentiation can be computed in advance during idle times.

In our conference paper[34], the code for the state-of-the-art comparison protocol based on DGK homomorphic encryption scheme have been used to compare the fact computation cost of the solution based on DGK and our solution based on symmetric commutative encryption scheme. It was written in C/C++ by Veugen et al.[32], which can be publicly available (http://cys.ewi.tudelft.nl/content/secure-comparison-protocols-semi-honest-model). But their code includes the Paillier computations function and implemented based on MPIR (Multiple Precision Integers and Rationals) Library. In this paper, in order to evaluate more accurately and fairly the fact computation cost of the solution based on DGK and our solution based on symmetric commutative encryption scheme, we implement our Protocol 4 and the state-of-the-art comparison protocol based on DGK

TABLE 4 Comparison in Computation Cost

Solution	Alice's computation	Bob's computation	Total computation
Lin et al.[29]	$3n\log(p)$	$2n\log(p)+4n-6$	$5n\log(p)+4n-6$
Lin et al. with hash[29]	$3n\log(p)$	$2n\log(p)+2n$	$5n\log(p)+2n$
DGK[33,19,32]	$nt/2+$ $n(l+2t+1)$	$4nt+4nl+2l+2t+$ $19n-12$	$6.5nt+$ $5nl+2l+2t+20n-12$
DGK with pre-computation[33,19,32]	$nt/2+$ $n(l+1)$	$4nl+2l-2t+$ $19n-12$	$0.5nt+$ $5nl+2l+20n-2t-12$
Ours	$2nt+n$	$2nt+n$	$4nt+n$
Ours with pre-computation	$nt+n$	$nt+n$	$2nt+n$

1. n represents the bit length of input.
2. p represents a modulus of G_p.
3. t represents the bit size of discrete log key or a bit length of random integer.

TABLE 5 Security parameter sizes for medium-term security

Security length	asymmetric key	discrete log key	discrete log group	hash
112	2048	224	2048	SHA-224

TABLE 6 Comparison results of processing time (ms) for a 24-bit and 200-bit inputs and medium-term security

solutions	our solution	our solution with pre-computations	Damgard's solution	Damgard's solution with pre-computations
24-bit	33.2	16.5	52.5†	8.9‡
220-bit	298.9	148.5	541.1	142.9

†Reported by our own code rather than 132.52s reported in the paper[34] with the code including Paillier computations written by Veugen.
‡Reported by our own code rather than 39.99s reported in the paper[34] with the code including Paillier computations written by Veugen.

homomorphic encryption scheme using Python 2.7 based on Charm and GNU Multiple Precision Arithmetic Library (GMP) 6.0.0. The experiments also are conducted on Windows 10 System with an Intel i5-4690 Processor at 3.50GHz and 8GB RAM.

For a 24-bit input and medium-term security in Table 5, the processing time has also been presented in Table 6 under the assumption that h^r of the DGK encryption function can be pre-computed during idle times. As evident from the Table 6, it can be observed that our solution outperforms Damgard's solution without pre-computations in terms of processing time, though our solution with pre-computations cannot outperform the state-of-the-art comparison protocol with pre-computations for a 24-bit input. However, for a 220-bit input, our solution can not only outperform Damgard's solution without pre-computations, but also compare well with the state-of-the-art comparison protocol with pre-computations.

In fact, our approach based on asymmetric commutative encryption scheme is a general solution to millionaires' problem. Thus, one of the further work is to employ the other asymmetric commutative encryption scheme, such as the one based on the elliptic-curve DDH (ECDDH) assumption, to improve the computation and communication cost in our solution.

6 | CONCLUSIONS

Shundong's symmetric cryptographic solution to millionaires' problem is proposed based on a private set-inclusion problem. We find that some drawbacks in the protocol to a private set set-inclusion problem and propose some simply solution to solve them. Furthermore, even if we postulate that symmetric encryption solution to the set-inclusion problem is perfect, their symmetric solution to millionaires' problem also exhibits some important security drawbacks that have been analyzed and proven in this paper. This paper deals with Shundong's original work and introduces a new solution based on the Decisional Diffie-Hellman assumption and the set intersection problem. Our solution based on the Decisional Diffie-Hellman assumption is an asymmetric commutative encryption scheme. To reduce the computation cost of modular multiplications, we also use two special encodings, 0-encoding and 1-encoding, to reduce the scale of the set intersection problem. To compare the fact computation cost of the solution based on symmetric commutative encryption scheme and asymmetric commutative encryption scheme, we implement XOR operation in Shundong's protocol and our protocol. It is found that Shundong's protocol is not more efficient than our protocol when the size of the input is large. And experimental results show that our solution is also more efficient than the state-of-the-art solution without pre-computation and can also compare well with the state-of-the-art comparison protocol while the bit length of private inputs is large enough.

References

1. Yao Andrew C.. Protocols for Secure Computations. In: SFCS '82:160–164IEEE Computer Society; 1982; Washington, DC, USA.
2. Du Wenliang, Atallah Mikhail J.. Privacy-Preserving Cooperative Scientific Computations. In: CSFW '01:273–282IEEE Computer Society; 2001; Washington, DC, USA.
3. Xiong H., Zhang E.P., Chim T.W., Yiu S.M., Hui L. C K. Weighted average problem revisited under hybrid and malicious model. In: :677-682; 2012.
4. Agrawal Rakesh, Srikant Ramakrishnan. Privacy-preserving Data Mining. In: SIGMOD '00:439–450ACM; 2000; New York, NY, USA.
5. Lindell Yehuda, Pinkas Benny. Privacy preserving data mining. In: :36–54Springer; 2000.
6. Jha Somesh, Kruger Luis, McDaniel Patrick. Privacy Preserving Clustering. In: ESORICS'05:397–417Springer-Verlag; 2005; Berlin, Heidelberg.
7. Cachin Christian. Efficient Private Bidding and Auctions with an Oblivious Third Party. In: CCS '99:120–127ACM; 1999; New York, NY, USA.
8. Shih Dong-Her, Huang Hsin-Yi, Yen David C.. A Secure Reverse Vickrey Auction Scheme with Bid Privacy. *Information Sciences.* 2006;176(5):550–564.
9. Bogetoft Peter, Christensen Dan Lund, Damgård Ivan, et al. Secure multiparty computation goes live. In: Springer 2009 (pp. 325–343).
10. Atallah Mikhail J., Du Wenliang. Secure Multi-party Computational Geometry. In: WADS '01:165–179Springer-Verlag; 2001; London, UK, UK.
11. Yang Bo, Sun AD, Zhang WZ. Secure two-party protocols on planar circles. *Journal of Information.* 2011;8(1):29–40.
12. Yang B, Shao ZY, Zhang WZ. Secure two-party protocols on planar convex hulls. *Journal of Information and Computational Science.* 2012;9(4):915–929.
13. Liu Liang, Wu Chunying, Li Shundong. Two privacy-preserving protocols for point-curve relation. *Journal of Electronics (China).* 2012;29(5):422–430.

14. YANG Bo, YANG Chung-Huang, YU Yong, XIE Dan. A Secure Scalar Product Protocol and Its Applications to Computational Geometry. *Journal of Computers.* 2013;8(8):2018–2026.

15. Freedman MichaelJ., Nissim Kobbi, Pinkas Benny. Efficient Private Matching and Set Intersection. In: Cachin Christian, Camenisch JanL., eds. *Advances in Cryptology - EUROCRYPT 2004*, Lecture Notes in Computer Science, vol. 3027: Springer Berlin Heidelberg 2004 (pp. 1-19).

16. Xie Qi, Hengartner Urs. Privacy-preserving matchmaking for mobile social networking secure against malicious users. In: :252–259IEEE; 2011.

17. Dachman-Soled Dana, Malkin Tal, Raykova Mariana, Yung Moti. Efficient Robust Private Set Intersection. *Int. J. Appl. Cryptol..* 2012;2(4):289–303.

18. Du Wenliang, Han Yunghsiang S, Chen Shigang. Privacy-preserving multivariate statistical analysis: Linear regression and classification. In: :222–233Lake Buena Vista, Florida; 2004.

19. Erkin Zekeriya, Franz Martin, Guajardo Jorge, Katzenbeisser Stefan, Lagendijk Inald, Toft Tomas. Privacy-preserving face recognition. In: :235–253Springer; 2009.

20. Xia Zhihua, Xiong Neal N, Vasilakos Athanasios V, Sun Xingming. EPCBIR: An efficient and privacy-preserving content-based image retrieval scheme in cloud computing. *Information Sciences.* 2016;.

21. Abduljabbar Zaid Ameen, Jin Hai, Ibrahim Ayad, et al. Secure biometric image retrieval in IoT-cloud. In: :1–6IEEE; 2016.

22. Zhang L., Jung T., Liu K., et al. PIC: Enable Large-scale Privacy Preserving Content-based Image Search on Cloud. *IEEE Transactions on Parallel and Distributed Systems.* 2017;PP(99):1-1.

23. Tonyali Samet, Akkaya Kemal, Saputro Nico, Uluagac A. Selcuk, Nojoumian Mehrdad. Privacy-preserving protocols for secure and reliable data aggregation in IoT-enabled Smart Metering systems. *Future Generation Computer Systems.* 2017;.

24. Rane Shantanu, Boufounos Petros T. Privacy-preserving nearest neighbor methods: Comparing signals without revealing them. *IEEE Signal Processing Magazine.* 2013;30(2):18–28.

25. Lagendijk R. L., Erkin Z., Barni M.. Encrypted signal processing for privacy protection: Conveying the utility of homomorphic encryption and multiparty computation. *IEEE Signal Processing Magazine.* 2013;30(1):82-105.

26. Garg Sanjam, Mohassel Payman, Papamanthou Charalampos. TWORAM: Efficient Oblivious RAM in Two Rounds with Applications to Searchable Encryption. In: :563–592Springer-Verlag New York, Inc.; 2016; New York, NY, USA.

27. Rindal Peter, Rosulek Mike. Faster malicious 2-party secure computation with online/offline dual execution. In: :297–314; 2016.

28. Ioannidis Ioannis, Grama Ananth. An efficient protocol for Yao's millionaires' problem. In: :6–9IEEE; 2003.

29. Lin Hsiao-Ying, Tzeng Wen-Guey. An efficient solution to the millionaires' problem based on homomorphic encryption. In: :456–466Springer; 2005.

30. Gai K., Qiu M.. Blend Arithmetic Operations on Tensor-based Fully Homomorphic Encryption Over Real Numbers. *IEEE Transactions on Industrial Informatics.* 2017;:1-1.

31. Shundong Li, Daoshun Wang, Yiqi Dai, Ping Luo. Symmetric cryptographic solution to Yao's millionaires' problem and an evaluation of secure multiparty computations. *Information Sciences.* 2008;178(1):244–255.

32. Veugen T., Blom F., Hoogh S. J. A., Erkin Z.. Secure Comparison Protocols in the Semi-Honest Model. *IEEE Journal of Selected Topics in Signal Processing.* 2015;9(7):1217-1228.

33. Damgard Ivan, Geisler Martin, Kroigard Mikkel. Homomorphic Encryption and Secure Comparison. *International Journal of Applied Cryptography.* 2008;1(1):22–31.

34. Liu Meng, Nanda Priyadarsi, Zhang Xuyun, Yang Chi, Yu Shui, Li Jianxin. Asymmetric Commutative Encryption Scheme Based Efficient Solution to the Millionaires' Problem. In: :990–995IEEE; 2018.

35. Fagin Ronald, Naor Moni, Winkler Peter. Comparing Information Without Leaking It. *Commun. ACM.* 1996;39(5):77–85.

36. Agrawal Rakesh, Evfimievski Alexandre, Srikant Ramakrishnan. Information sharing across private databases. In: :86–97ACM; 2003.

37. Shundong Li, Tiange Si, Yiqi Dai. Secure multi-party computation of set-inclusion and graph-inclusion. *Journal of Computer Research and development.* 2005;10:1647–1653.

38. Blake IanF., Kolesnikov Vladimir. Strong Conditional Oblivious Transfer and Computing on Intervals. In: Lecture Notes in Computer Science, vol. 3329: Springer Berlin Heidelberg 2004 (pp. 515-529).

39. Goldreich Oded. *Foundations of Cryptography: Volume 2, Basic Applications.* Cambridge university press; 2009.

40. Akinyele Joseph A., Garman Christina, Miers Ian, et al. Charm: a framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering.* 2013;3(2):111-128.

41. GNU . The GNU Multiple Precision Arithmetic Library .

42. Pollard John M.. Monte Carlo methods for index computation mod *p*. *Mathematics of Computation.* 1978;32:918–924.

43. Wenbo Mao. Modern cryptography: theory and practice. *Publisher: Prentice Hall PTR, Copyright: Hewlett Packard.* 2004;.

44. Damgard Ivan, Geisler Martin, Kroigard Mikkel. A correction to 'efficient and secure comparison for on-line auctions'. *International Journal of Applied Cryptography.* 2009;1(4):323–324.

45. Shih Dong-Her, Huang Hsin-Yi, Yen David C. A secure reverse Vickrey auction scheme with bid privacy. *Information Sciences.* 2006;176(5):550–564.

46. Bogetoft Peter, Christensen Dan Lund, Damgård Ivan, et al. In: Dingledine Roger, Golle Philippe, eds. *Financial Cryptography and Data Security*, Berlin, Heidelberg: Springer-Verlag 2009 (pp. 325–343).

47. Xie Qi. Privacy-Preserving Interest Matching for Mobile Social Networking. Master's thesisUniversity of Waterloo2010.

48. Wang Yong, Hou Jie, Tan Yuan-wei, Nie Xiao. A Recommendation-based Matchmaking Scheme for Multiple Mobile Social Networks against Private Data Leakage. *Procedia Computer Science.* 2013;17:781–788.

49. Blake Ian F, Kolesnikov Vladimir. Strong conditional oblivious transfer and computing on intervals. In: Springer 2004 (pp. 515–529).

50. Mao Wenbo. *Modern cryptography: theory and practice.* Prentice Hall Professional Technical Reference; 2003.

51. Gai Keke, Qiu Meikang, Xiong Zenggang, Liu Meiqin. Privacy-preserving multi-channel communication in Edge-of-Things. *Future Generation Computer Systems.* 2018;85:190 - 200.

How to cite this article: M. Liu, P. Nanda, X. Zhang, C. Yang, S. Yu, and J. Li (2018), Efficient Solution to the Millionaires' Problem Based on Asymmetric Commutative Encryption Scheme, *Computational Intelligence*, *2017;00:1–6*.