Hindawi
Security and Communication Networks
Volume 2022, Article ID 6923610, 11 pages
https://doi.org/10.1155/2022/6923610

Research Article

Research on the Millionaires' Problem under the Malicious Model Based on the Elliptic Curve Cryptography

Xin Liu,[1] Yang Xu,[1] Gang Xu ,[2,3] and Baoshan Li[1]

[1]*School of Information Engineering, Inner Mongolia University of Science and Technology, Baotou 014010, China*
[2]*School of Information Science and Technology, North China University of Technology, Beijing 100144, China*
[3]*Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing Jiaotong University, Beijing 100144, China*

Correspondence should be addressed to Gang Xu; gx@ncut.edu.cn

Received 22 October 2021; Accepted 10 February 2022; Published 23 March 2022

Academic Editor: Xin-Yi Huang

With the rapid development of blockchain, big data, cloud computing, and artificial intelligence, the security of multisource data collaborative computing has become increasingly prominent. Secure multiparty computing has become the core technology of privacy collaborative computing. Millionaires' problem is the cornerstone of secure multiparty computation. Firstly, this paper proposes a 0-1 coding rule, which is used to solve the millionaires' problem under the semihonest model. Aiming at the possible malicious behaviors of the protocol under the semihonest model, the millionaires' problem protocol under the malicious model based on the elliptic curve cryptography is designed by using cryptographic tools such as the zero-knowledge proof and the cut-choose method. This protocol not only can effectively solve the millionaires' problem but also can safely and effectively prevent malicious behaviors. Meanwhile, the security ordering designed by the protocol can be effectively applied to a quality evaluation in the blockchain.

1. Introduction

Secure multiparty computation (SMC) is the core technology to achieve collaborative computing for the privacy of multisource data in recent years. The idea of SMC is proposed by Professor Yao Qizhi in 1982 [1], and then Goldreich [2, 3] began to do more in-depth research on SMC. SMC has been widely used in the blockchain [4–6], data mining [7–9], privacy computing [10, 11], medical [12, 13].

The millionaires' problem is one of the most classic problems in SMC. Many cryptographers have been working on it. Reference [14] presents a protocol for solving the millionaires' problem based on the exchange cryptosystem, oblivious transfer method. Based on the Goldwasser–Micali (GM) cryptography, reference [15] proposed a protocol to solve the problem of socialist millionaires. Reference [16] proposed a protocol to solve the problem of the millionaires' problem by using the shift registers and the property of probability encryption. Reference [17] presents a protocol based on the Paillier cryptosystem. The existing schemes have the disadvantage of inefficiency, and most of them are only suitable for the semihonest model and cannot resist malicious attacks. To solve the above problems, this paper studies the millionaires' problem under the malicious model in depth and presents the millionaires' problem protocol based on the elliptic curve cryptography.

Elliptic curve cryptography (ECC) is a traditional encryption method that has the advantage of high computational efficiency and is based on the elliptic curve discrete logarithm problem, and it has been widely used because of its short key [18–20]. Using ECC, the protocol designed in this paper has more efficient operation efficiency and security. The main contributions are as follows:

(1) First, a 0-1 encoding rule for ECC is proposed, and then a millionaires' problem protocol under the semihonest model is designed

(2) With the help of some cryptographic tools such as the zero-knowledge proof and cut-choose method, a

protocol is designed to resist the attacks of malicious opponents

(3) Finally, an ideal-practical example method is used to prove the security of the protocol under the malicious model

2. Preliminary Knowledge

2.1. Elliptic Curve Cryptography. Elliptic curve cryptography (ECC) is a public-key cryptosystem based on discrete logarithmic problems of point groups of elliptic curves. For example, an elliptic curve is defined as $y^2 = x^3 - x$, two points P and Q on the curve, a straight line through P and Q, an intersecting elliptic curve at R' point, and a line perpendicular to X axis through R' point. An intersecting elliptic curve at another point R is defined as $P + Q = R$, as shown in Figure 1.

When $P = Q$, the tangent of the point P intersects R', and then the point R' makes a straight line perpendicular to the X-axis, intersecting the elliptic curve at another point R. When k identical P are added, they are counted as kP, such as $P + P + P + P = P + 3P = 4P$. Elliptic curves make use of the mathematical problem of discrete logarithm in the above operations, that is, when $K = kP$, P and K are known, and K is easily obtained. But it is very difficult to find P and K when K is known.

In cryptography systems, if there is an elliptic curve $E_p(a, b)$ and a base point G, a random number k is generated as the private key, and the datum is computed k times to get the public key $K = kG$. In ECC, the private key k is easy to get the public key K; the public key K cannot get the private key k.

Compared with the traditional public key algorithms, elliptic curve cryptography has the following advantages [19, 20]:

(1) higher security performance. For example, 160 bit ECC has the same security strength as 1024 bit RSA and DSA algorithms.

(2) small amount of computation and fast processing speed. ECC is much faster than RSA and DSA in the processing speed of private key (decryption and signature).

(3) The storage space occupation is small. Compared with RSA and DSA, the key size and system parameters of ECC are much smaller, so the storage space occupation is much smaller.

(4) ECC has a wide application prospect because of its low bandwidth requirements.

2.1.1. ECC Encryption

(1) Elliptic curve $E_p(a, b)$, a base point G, private key k, and public key $K = kG$

(2) Encode a plaintext a to a point M on the elliptic curve $E_p(a, b)$ and select a random number $r < n$ (where n is the order of base point G)

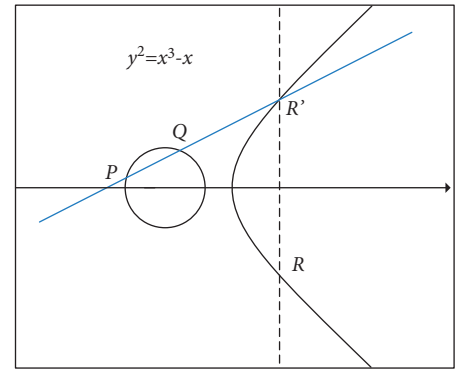

Figure 1: Elliptic curve operation, defined $P + Q = R$.

(3) Encryption: $C_1 = M + rK$, $C_2 = rG$

(4) Decryption: $C_1 - kC_2 = M + rK - k(rG) = M + r(K - kG) = M$

2.1.2. Additive Homomorphism of ECC

(1) Encryption: encode a plaintext $a_i (0 < i \le m)$ onto a point M_i of the elliptic curve $E_p(a, b)$, use the private key k to generate the public key $K = kG$, select a random number $r_i < n$, and calculate $C_{1i} = M_i + rK$ and $C_{2i} = rG$

(2) Addition operation: all ciphertexts are added to obtain ciphertexts $(\sum_{i=1}^{m} C_{1i} \sum_{i=1}^{m} C_{2i})$, where $\sum_{i=1}^{m} C_{1i} = C_{11} + C_{12} + \cdots C_{1m}$ and $\sum_{i=1}^{m} C_{2i} = C_{21} + C_{22} + \cdots C_{2m}$

(3) Decryption: compute $\sum_{i=1}^{m} C_{1i} - k \sum_{i=1}^{m} C_{2i} = \sum_{i=1}^{m} M_i + K \sum_{i=1}^{m} r_i - kG \sum_{i=1}^{m} r_i = \sum_{i=1}^{m} M_i$ and finally decode $\sum_{i=1}^{m} M_i$ to obtain the plaintext $\sum_{i=1}^{m} a_i$

2.2. Zero-Knowledge Proof. Zero-knowledge proof [21] means that the prover can make the verifier believe that a conclusion is correct without providing any useful information to the verifier. The prover proves to the verifier and makes him believe that he knows or owns a certain message, but the certification process cannot disclose any information about the proved message to the verifier. A large number of facts have proved that the zero-knowledge proof is very useful in cryptography. If the zero-knowledge proof can be used for verification, many problems can be effectively solved.

2.3. Cut-Choose Method. In cryptography, we encrypt the information that can be sent into n messages using n different random numbers. The receiver selects $n/2$ of them to verify their correctness and then selects one of the remaining $n/2$ for the remaining protocol steps. The cut-choose method can minimize the malicious input probability of the protocol and make the transmission of information more secure [22].

2.4. Security Definition of a Protocol under the Malicious Model. To prove that a protocol is secure under the malicious model, it must satisfy the security definition under the

malicious model. If the actual protocol achieves the same security as the ideal protocol, then the protocol is secure [3].

Alice owns x, and Bob owns y. They compare by calculating function $f(x,y) = (f_1(x,y), f_2(x,y))$ with a trusted third party (TTP). At the end of the protocol, both parties get $f_1(x,y)$ and $f_2(x,y)$ without leaking x and y. The ideal model is as follows:

(1) The honest participant always provides x or y to the TTP, while the malicious participant may decide not to execute the protocol based on x or y or provide a false input x' or y' to the TTP when the protocol is executed.

(2) TTP sends the result to Alice. After TTP gets input pair (x,y), $f(x,y)$ is calculated, and $f_1(x,y)$ is sent to Alice; otherwise, the special symbols $\perp$ are sent to Alice.

(3) TTP sends the result to Bob, and if Alice is a malicious participant, it may no longer contact with TTP after receiving $f_1(x,y)$. In this case, TTP sends Bob a special symbol $\perp$; otherwise, TTP sends $f_2(x,y)$ to Bob.

The ideal protocol is the safest protocol because participants cannot get any information except their $f_i(x,y)$ from TTP. If an actual protocol achieves the same security as the ideal protocol, we say that the actual protocol is secure.

If the participant in the ideal model has an auxiliary information z and the process of calculating $F(x,y)$ in combination with policy $\overline{B}$ is $\mathrm{IDEAL}_{F,\overline{B}(z)}(x,y)$, it is defined as the adversary evenly choosing a random number r and to make $\mathrm{IDEAL}_{F,\overline{B}(z)}(x,y) = \gamma(x,y,z,r)$, where $\gamma(x,y,z,r)$ is defined as follows (note: if both parties under the malicious model are malicious, it is impossible to design an SMC protocol; we do not consider this case):

(1) If Alice is honest, there is $\gamma(x,y,z,r) = (f_1(x,y'), B_2(y,z,r,f_2(x,y')))$, where $y' = B_2(y,z,r)$.

(2) If Bob is honest:

$$\gamma(x,y,z,r) = \begin{cases} \left(B_1\left(x,z,r,f_1(x\prime,y),\perp\right),\perp\right), & \text{if}: B_1(x,z,r,f_1(x',y)) = \perp, \\ \left(B_1\left(x,z,r,f_1(x\prime,y)\right),f_2(x\prime,y)\right), & \text{otherwise.} \end{cases} \tag{1}$$

In both cases, $x' = B_1(x,z,r)$.

Definition 1. Security for the malicious model.

If, in the ideal model, an acceptable policy pair $\overline{A} = (A_1, A_2)$ in the actual protocol can be found, there is an acceptable policy pair $\overline{B} = (B_1, B_2)$, such that

$$\left\{\mathrm{IDEAL}_{F,\overline{B}(z)}(x,y)\right\}_{x,y,z} \stackrel{c}{\equiv} \left\{\mathrm{REAL}_{\overline{A}(z)}(x,y)\right\}_{x,y,z}. \tag{2}$$

So this protocol can calculate F securely.

3. The Protocol under the Semihonest Model

3.1. Solution Ideas. Alice owns x, and Bob owns y. Alice and Bob want to compare the relationship: $x > y$, $x = y$, $x < y$, while they do not want to leak their x and y, respectively. The solution is to code x and y as a set consisting of 1 and 0 and use ECC to design an efficient protocol.

The 0-1 encoding rule: encode x into a set $X = (a_1, a_2, \ldots, a_m)$, where $a_1 < a_2 < \cdots < a_m$ and

$$a_i = \begin{cases} 1, & i = x; \\ 0, & i \neq x. \end{cases} \tag{3}$$

The comparison rule: based on the position of y in X: if $x > y$, then $\sum_{i=1}^{y-1} a_i + \sum_{i=1}^{y} a_i = 0$; if $x = y$, then $\sum_{i=1}^{y-1} a_i + \sum_{i=1}^{y} a_i = 1$; and if $x < y$, then $\sum_{i=1}^{y-1} a_i + \sum_{i=1}^{y} a_i = 2$. Define the following formula to judge the relationship between x and y:

$$P(x,y) = \begin{cases} 0, & x > y; \\ 1 & x = y; \\ 2, & x < y. \end{cases} \tag{4}$$

For example: Alice's data is 5, which is encoded into a new set $X = (0,0,0,0,1,0,0)$, and Bob calculates with three different data $y = 2, 5, 7$. This is shown in Table 1.

3.2. Specific Protocol. Alice owns data x, and Bob owns data y, and both parties compute $P(x,y)$ securely to determine the relationship. Using the above 0-1 encoding rule, Algorithm 1 under the semihonest model is designed based on ECC homologous encryption.

Algorithm 1 is secure under the semihonest model, but if one of Alice and Bob is a malicious participant, the protocol is no longer secure. The following section will improve the protocol to make it safe and feasible under the malicious model.

4. The Protocol under the Malicious Model

4.1. Solution Ideas. Firstly, we analyze the possible malicious attacks in Algorithm 1. Then the solutions to these malicious attacks are proposed. Finally, the possible malicious attacks cannot be implemented or found when they are committed. The following malicious attacks may exist in Algorithm 1 as follows:

(1) In Algorithm 1, Alice has both the public key K and the private key k, but Bob only has the public key K, so the final result can only be calculated unilaterally

Table 1: The 0-1 encoding. Data comparison results of Alice and Bob.

Alice's data	New set for encoding	Bob's data	Calculate $w = \sum_{i=1}^{y-1} a_i + \sum_{i=1}^{y} a_i$	Comparison results
5	$X = (0,0,0,0,1,0,0)$	2	$w = 0 + 0 = 0$	$x > y$
		5	$w = 0 + 1 = 1$	$x = y$
		7	$w = 1 + 1 = 2$	$x < y$

Input: Alice owns data x, and Bob owns data y.
Output: $P(x, y)$.
(1) Alice encodes x into a set $X = (a_1, a_2, \ldots, a_m)$, where $a_1 < a_2 < \cdots < a_m$ and $a_i = \begin{cases} 1, & i = k; \\ 0, & i \neq k. \end{cases}$.
(2) Alice chooses an elliptic curve E_p, the base point G, and the private key k; then calculates $kG = K$ as the public key K; and publishes the public key K and the base point G.
(3) Alice encodes the plaintext $X = (a_1, a_2, \ldots, a_m)$ one by one onto point M_i $(1 \le i \le m)$ on the elliptic curve E_p (the encoding method is not unique [18], which is not discussed here). She chooses m random numbers r_i and encrypts each element M_i one by one using the public key K of ECC, that is, $E(M_i) = (C_{1i}, C_{2i})$, $C_{1i} = M_i + r_i K + C_{2i} = r_i G$. She gets $E(X) = (E(M_1), E(M_2), \ldots, E(M_m))$, which is sent to Bob.
(4) Bob calculates $E(W) = (C_1, C_2)$ based on the y position of data in $E(X)$, where $C_1 = \sum_{i=1}^{y-1} C_{1i} + \sum_{i=1}^{y-1} C_{1i}$ and $C_2 = \sum_{i=1}^{y-1} C_{2i} + \sum_{i=1}^{y} C_{2i}$. He sends $E(W) = (C_1, C_2)$ to Alice.
(5) Alice decrypts (W) with the private key k to get W, where $C_1 - kC_2 = \sum_{i=1}^{y-1} M_i + \sum_{i=1}^{y} M_i = W$, and decodes the point W to get $w = \sum_{i=0}^{y-1} a_i + \sum_{i=0}^{y} a_i$. If $w = 0$, $x > y$; if $w = 1$, $x = y$; and if $w = 2$, $x < y$. Alice tells Bob the result.
The protocol ends.

Algorithm 1: Judgement under the semihonest model.

by Alice, which is unfair to Bob. (2) In steps 3 and 4, if the ciphertext sent by Alice and Bob to each other is wrong so that neither party can get the correct result. (3) In step 5, Alice tells Bob the wrong result after decrypting, which leads to a wrong conclusion for Bob. For the above malicious attacks, a new protocol must be designed to find or render them impossible to implement. (Note: Before designing the protocol under the malicious model, we need to be clear that some malicious behaviors cannot be prevented in the ideal protocol. For example, if you enter wrong inputs in the ideal model, no matter how you detect and verify, you cannot get the correct results; similarly, if you refuse to carry out the protocol, we cannot get the results, either. Therefore, we will not consider the following behaviors when designing the protocol under the malicious model: (1) refusing to carry out the protocol, (2) inputting false data, and (3) one party terminating the protocol after obtaining the information he wants to prevent other participants from carrying out the protocol.)

To design a secure, fair, and correct protocol under the malicious model, the solution is to use cryptographic tools such as the zero-knowledge proof and cut-choose method to prevent malicious attacks that may exist in Algorithm 1. The final results are calculated by both parties at the same time.

4.2. Specific Protocol. Based on the malicious attacks that may occur in Algorithm 1 under the semihonest model, we use the above 0-1 encoding rule to design the millionaires' problem algorithm under the malicious model using the zero-knowledge proof and cut-choose method. The framework of Algorithm 2 under the malicious model is outlined in Algorithm 3.

A specific protocol is as follows:

4.3. Correctness Analysis

(1) The steps and positions for both Alice and Bob to execute the protocol in Algorithm 2 are identical, so we only demonstrate the possible malicious behaviors of Alice. The security analysis of the protocol is as follows:

In step (5), if Alice selects a_i that is the wrong random number, Bob happens not to choose the wrong random number a_i out of $m/2$ selected, that is, no wrong random number is detected, but in the following step (7), it happens to be selected by Bob, and Bob calculates the wrong result. The probability of success is analyzed as follows:

① If Alice uses the above method to commit a malicious attack, the most likely scenario for successful execution of such malicious attacks is that Alice mixes one wrong a_i in m random a_i, which maximizes the likelihood that the malicious attack will succeed. The probability of deception success in this case is $1/m$.

② If $m = 20$, Alice mixes one wrong a_i in m random a_i. The probability of deception success in this case is $C_{19}^{10}/C_{20}^{10} \times 1/10 = 1/200$, but if Alice mixes 10 wrong a_i in m random a_i, the probability of deception success in this case is $C_{19}^{10}/C_{20}^{10} \times 1/2 = 2.7 \times 10^{-7}$, in which case the probability of success is even smaller or negligible.

Input: Alice owns x, and Bob owns y.
Output: $P(x, y)$.
Prepare:

(1) Alice and Bob jointly select an elliptic curve E_p and a base point G. Alice and Bob separately select their own private key k_1, k_2 ($k_1, k_2 > 0$). Then Alice and Bob calculate their public keys $K_1 = k_1 G$ and $K_2 = k_2 G$ and $u = aK_1$ and $v = bK_2$, respectively. Finally, Alice and Bob exchange (K_1, u) and (K_2, v).
Alice and Bob construct their own new sets $X = (a_1, a_2, \ldots, a_m)$ and $Y = (b_1, b_2, \ldots, b_m)$ through x and y, where: $a_i = \begin{cases} 1, & i = x; \\ 0, & i \neq x. \end{cases} b_i = \begin{cases} 1, & i = y; \\ 0, & i \neq y. \end{cases}$.
Start:

(1) Alice encodes the plaintext $X = (a_1, a_2, \ldots, a_m)$ onto the point M_i^a ($1 \le i \le m$) of the elliptic curve $E_p(a, b)$; selects m random numbers r_i^a; encrypts each element M_i^a one by one with the public key K_1, that is, calculates the $E(M_i^a) = (C_{1i}^a, C_{2i}^a)$, where: $C_{1i}^a = M_i^a + r_i^a K_1$ and $C_{2i}^a = r_i^a G$; obtains $E(X) = (E(M_1^a), E(M_2^a), \ldots, E(M_m^a))$; and finally, sends $E(X)$ to Bob.
(2) Bob encodes the plaintext $Y = (b_1, b_2, \ldots, b_m)$ onto the point M_i^b ($1 \le i \le m$) of the elliptic curve $E_p(a, b)$; selects m random numbers r_i^b; encrypts each element M_i^b one by one by using the public key K_2, that is, calculates the $E(M_i^b) = (C_{1i}^b, C_{2i}^b)$, where: $C_{1i}^b = M_i^b + r_i^b K_2$ and $C_{2i}^b = r_i^b G$; obtains $E(Y) = (E(M_1^b), E(M_2^b), \ldots, E(M_3^b))$; and finally, sends $E(Y)$ to Alice.
(3) Alice calculates $E(Q) = (C_1, C_2)$ according to the position of data x in $E(Y)$, where $C_1 = \sum_{i=1}^{x-1} C_{1i}^b + \sum_{i=1}^{x} C_{1i}^b$ and $C_2 = \sum_{i=1}^{x-1} C_{2i}^b + \sum_{i=1}^{x} C_{2i}^b$, and sends $E(Q)$ to Bob.
Bob calculates $E(W) = (C_1', C_2')$ according to the position of data y in $E(X)$, where $C_1' = \sum_{i=1}^{y-1} C_{1i}^a + \sum_{i=1}^{y} C_{1i}^a$ and $C_2' = \sum_{i=1}^{y-1} C_{2i}^a + \sum_{i=1}^{y} C_{2i}^a$, and sends $E(W)$ to Alice.
(4) Alice decrypts $E(W)$ using the private key k_1, that is, calculates $C_1 - k_1 C_2 = \sum_{i=0}^{x-1} M_i^b + \sum_{i=0}^{x} M_i^b = W$ to obtain W point. Bob decrypts $E(Q)$ using the private key k_2, that is, calculates $C_1' - k_2 C_2' = \sum_{i=0}^{y-1} M_i^a + \sum_{i=0}^{y} M_i^a = Q$ to obtain point Q.
(5) Alice selects m random numbers d_i ($0 \le i \le m$) to calculate $(c_{1a}^i, c_{2a}^i) = (d_i W + K_1,\ W + d_i W + aG)$. Bob selects m random numbers f_i ($0 \le i \le m$) to calculate $(c_{1b}^i, c_{2b}^i) = (f_i Q + K_2, Q + f_i Q + bG)$. Finally, Alice and Bob exchange (c_{1a}^i, c_{2a}^i) and (c_{1b}^i, c_{2b}^i).
(6) With the help of the cut-choose method, Alice randomly selects the $m/2$ groups from the m groups (c_{1b}^i, c_{2b}^i) sent by Bob and publishes it and requires Bob to publish the corresponding $f_i Q$. Alice verifies: $f_i Q + K_2 = c_{1b}^i$. If the verification is passed, they continue the protocol or else terminate.
Bob randomly selects the $m/2$ groups from the m group (c_{1a}^i, c_{2a}^i) sent by Alice and publishes it and requires Alice to publish the corresponding $d_i W$. Bob verifies: $d_i Q + K_1 = c_{1a}^i$. If the verification is passed, they continue the protocol or else terminate.
(7) Alice and Bob randomly select one (c_{1b}^i, c_{2b}^i) and (c_{1a}^i, c_{2a}^i) from the remaining (c_{1b}^i, c_{2b}^i) and (c_{1a}^i, c_{2a}^i), respectively. Meanwhile, Alice selects two random numbers h and p_1, and Bob selects two random numbers l and p_2. Alice calculates
$c_b = h(c_{2b}^j - c_{1b}^j - W + K_2) = h(Q - W) + hlG$, $P_1 = p_1 G$, $\lambda_b = p_1 K_2$; Bob calculates
$c_a = l(c_{2a}^i - c_{1a}^i - Q + K_1) = l(W - Q) + hlG$, $P_2 = p_2 G$, and $\lambda_a = p_2 K_1$. Then Alice and Bob send $c_b + P_1$ and $c_a + P_2$ to each other.
(8) After both parties receive information from each other, Alice calculates $\omega_a = k_1(c_a + P_2)$ and $m_a = k_1 c_a$ and sends them to Bob. Bob calculates $\omega_b = k_2(c_b + P_1)$ and $m_b = k_2 c_b$ and sends them to Alice.
(9) Alice uses the zero-knowledge proof to verify that the m_b sent by Bob is correct, that is, to prove that Bob does get the m_b by multiplying his private key k_2 with his c_b, that is, to judge whether $m_b = \omega_b - \lambda_b$ is true. Bob uses the zero-knowledge proof to verify that the m_a sent by Alice is correct, that is, to prove that Alice does get the m_a by multiplying her private key k_1 with her c_a, that is, to judge whether $m_a = \omega_a - \lambda_a$ is true. The party who fails is malicious.
(10) Alice can get $k_2 h(Q - W)$ by calculating $m_b - hv$. If $k_2 h(Q - W) = 0$, then $Q = W$; Bob can get $k_1 l(W - Q)$ by calculating $m_a - lu$. If $k_1 l(W - Q) = 0$, then $Q = W$. If $Q = W$, it proves that the results required by both parties are correct and identical; otherwise, the protocol shall be terminated.
(11) Finally, Alice and Bob get $\sum_{i=0}^{x-1} b_i + \sum_{i=0}^{x} b_i = w$ and $\sum_{i=0}^{y-1} a_i + \sum_{i=0}^{y} a_i = q$ by decoding points W and Q, respectively. If $q, w = 0$, then $x > y$; if $q, w = 1$, then $x = y$; and if $q, w = 2$, then $x < y$.
The protocol ends.

Algorithm 2: Judgement under the malicious model.

Input: x: Alice's input; y: Bob's input; G: the base point of the elliptic curve E_p; *Co de*: encode inputs into a m degree 0-1 codes; k_1: Alice's private key; k_2: Bob's private key; E: encrypt. h, p_1: Alice's random number; and l, p_2: Bob's random number

(1) $K_1 = k_1 G K_2 = k_2 G$
(2) $u = aK_1 v = bK_2$
(3) $P_1 = p_1 G P_2 = p_2 G$
(4) $\lambda_b = p_1 K_2 \lambda_a = p_2 K_1$
(5) $\text{Code}(x) = X = (a_1, a_2, \ldots, a_m)$
(6) $C_{1i}^a = M_i^a + r_i^a K_1 C_{2i}^a = r_i^a G$
(7) $E(M_i^a) = (C_{1i}^a, C_{2i}^a)$
(8) $E(X) = (E(M_1^a), E(M_2^a), \ldots, E(M_m^a))$

Algorithm 3: Continued.

(9) $Co\ de(y) = Y = (b_1, b_2, \cdots, b_m)$
(10) $C_{1i}^b = M_i^b + r_i^b K_2 C_{2i}^b = r_i^b G$
(11) $E(M_i^b) = (C_{1i}^b, C_{2i}^b)$
(12) $E(Y) = (E(M_1^b), E(M_2^b), \ldots, E(M_3^b))$
(13) Exchange $(K_1, E(X), u)$, $(K_2, E(Y), v)$.
(14) $C_1 = \sum_{i=1}^{x-1} C_{1i}^b + \sum_{i=1}^{x} C_{1i}^b C_2 = \sum_{i=1}^{x-1} C_{2i}^b + \sum_{i=1}^{x} C_{2i}^b$
(15) $E(Q) = (C_1, C_2)$
(16) $C_2' = \sum_{i=1}^{y-1} C_{2i}^a + \sum_{i=1}^{y} C_{2i}^a$
(17) $E(W) = (C_1', C_2')$
(18) Exchange $E(Q)$, $E(W)$
(19) $D(E(W)) = WD(E(Q)) = Q$
(20) $(c_{1a}^i, c_{2a}^i) = (d_i W + K_1, W + d_i W + aG)$and $(c_{1b}^i, c_{2b}^i) = (f_i Q + K_2, Q + f_i Q + bG)$
(21) $((c_{1a}^i, c_{2a}^i), (c_{1b}^i, c_{2b}^i))$
(22) Alice verifies if $f_i Q + K_2 = c_{1b}^i$ and then continues or else terminates
(23) Bob verifies if $d_i W + K_1 = c_{1a}^i$ and then continues or else terminates
(24) $c_b = h(c_{2b}^j - c_{1b}^j - W + K_2) = h(Q - W) + hlG$
(25) $c_a = l(c_{2a}^i - c_{1a}^i - Q + K_1) = l(W - Q) + hlG$
(26) $m_a = k_1 c_a m_b = k_2 c_b$
(27) Exchange $(c_b + P_1, m_a)$, $(c_a + P_2, m_b)$
(28) $\omega_a = k_1(c_a + P_2)\omega_b = k_2(c_b + P_1)$
(29) Exchange ω_a, ω_b
(30) $m_b = \omega_b - \lambda_b m_a = \omega_a - \lambda_a$
(31) $m_b - hv \Longrightarrow k_2 h(Q - W)m_a - lu \Longrightarrow k_1 l(W - Q)$
(32) if $Q = W$, then
$D(W) = \sum_{i=0}^{x-1} b_i + \sum_{i=0}^{x} b_i = w$ and $D(Q) = \sum_{i=0}^{x} b_i + \sum_{i=0}^{y} a_i = q$
if $q, w = 0$, then $x > y$
else if $q, w = 1$, then $x = y$
else $x < y$
else terminate
Output: $P(x, y)$

Algorithm 3: Judgement under the malicious model.

③ If Alice mixes m random a_i with more than $m/2$ wrong random numbers, it will be discovered in the subsequent verification phase.

(2) In step (4), both Alice and Bob decrypt point W and point Q using their respective private keys k_1 and k_2 as follows:

$$\begin{aligned} C_1 - k_1 C_2 &= \sum_{i=0}^{x-1} M_i^b + \sum_{i=0}^{x} M_i^b + K_1 \sum_{i=1}^{x-1} r_i - k_1 G \sum_{i=1}^{x-1} r_i \\ &\quad + K_1 \sum_{i=1}^{x} r_i - k_1 G \sum_{i=1}^{x} r_i = \sum_{i=0}^{x-1} M_i^b + \sum_{i=0}^{x} M_i^b = W, \\ C_1' - k_2 C_2' &= \sum_{i=0}^{y-1} M_i^a + \sum_{i=0}^{y} M_i^a + K_2 \sum_{i=1}^{y-1} r_i - k_2 G \sum_{i=1}^{y-1} r_i \\ &\quad + K_2 \sum_{i=1}^{y} r_i - k_2 G \sum_{i=1}^{y} r_i = \sum_{i=0}^{y-1} M_i^a + \sum_{i=0}^{y} M_i^a = Q. \end{aligned} \tag{5}$$

(3) The (c_{1a}^i, c_{2a}^i) and $(c_{1\ d}^i, c_{2\ d}^i)$ published in step (5) do not leak any information because their own random numbers are added.

(4) In step (7), Alice and Bob calculate as follows:

$$\begin{aligned} c_b &= h\left(c_{2b}^j - c_{1b}^j - W + K_2\right) \\ &= h\left(Q + f_j Q + bG - f_j Q - K_2 - W + K_2\right) \\ &= h(Q - W) + hlG, \\ c_a &= l\left(c_{2a}^i - c_{1a}^i - Q + K_1\right) \\ &= l\left(W + d_i W + aG - d_i W - K_1 - Q + K_1\right) \\ &= l(W - Q) + hlG. \end{aligned} \tag{6}$$

Alice and Bob then send $c_b + P_1$ and $c_a + P_2$ to each other.

(5) In step (10), Alice and Bob get the right results.

Alice uses the zero-knowledge to prove that the m_b sent by Bob is correct; the result obtained by calculating $m_b - hv$ is correct, that is,

$$\begin{aligned} m_b - hv &= m_b - hlK_2 \\ &= k_2 c_b - hlk_2 G \\ &= k_2 h(Q - W) + k_2 hlG - hlk_2 G \\ &= k_2 h(Q - W). \end{aligned} \tag{7}$$

After Bob uses zero-knowledge to prove that the m_a sent by Alice is correct; the result obtained by calculating $m_a - bu$ is correct, that is:

$$\begin{aligned} m_a - lu &= m_a - hlK_1 \\ &= k_1 c_a - hlk_1 G \\ &= k_1 l(W - Q) + k_1 hlG - hlk_1 G \\ &= k_1 l(W - Q). \end{aligned} \tag{8}$$

(6) In step (11), Alice and Bob decode point $W = \sum_{i=0}^{x-1} M_i^b + \sum_{i=0}^{x} M_i^b$ and point $Q = \sum_{i=0}^{y-1} M_i^a + \sum_{i=0}^{y} M_i^a$ to get $\sum_{i=0}^{x-1} b_i + \sum_{i=0}^{x} b_i = w$ and $\sum_{i=0}^{y-1} a_i + \sum_{i=0}^{y} a_i = q$, respectively.

(7) The encoding of plaintexts on points W and Q in steps (1) and (2) and the decoding of points W and Q in step (11) can be referred to reference [23].

(8) The whole process does not leak any confidential information, and both parties can obtain results independently, avoiding unfairness caused by one party telling the other party the result.

(9) In many cases, the data range is known to all parties in reality. For example, if two students want to compare their grades, then the data range is $(a_1, a_2, \ldots a_{100})$, that is known to all parties; if two companies at the same level want to compare their assets, the data range may be $(a_{1M}, a_{2M}, \ldots, a_{100M})$, but a company' assets are often sparsely rather than densely distributed on the data range. Assets can only be a few scales, and the data range is very small. Therefore, they know the data range. The data range does not leak any information about its private data. Generally speaking, all the numbers compared in SMC are comparable. If these figures are comparable, both parties will know their scope. Ordinary companies will never compare their assets with Microsoft because they are not comparable. However, we have to say that although the data range is known, if the data range is large, the computational complexity of the protocol will be very high, so the protocol becomes impractical.

4.4. Security Proof. Algorithm 2 under the malicious model is proved as follows.

Definition 2. Algorithm 2 is secure under the malicious model.

Proof. This proving process borrows a trusted third party (TTP). We set the actual policy pair as $\overline{A} = (A_1, A_2)$, the ideal policy pair as $\overline{B} = (B_1, B_2)$, F as the output, and S as the message sequence received by A_2 in the zero-knowledge proof process. We want to prove that the security of the protocol under the malicious model is to prove that when Algorithm 2 is executed, the implementation of malicious behaviors in the actual protocol calculation will not affect the correct output, that is:

$$\left\{\mathrm{REAL}_{\overline{A}}(W, Q)\right\} = \{\mathrm{IDEAL}_{\overline{B}}(W, Q)\}. \tag{9}$$

In Algorithm 2, the malicious behaviors are not allowed for both Alice and Bob at the same time, so there are two scenarios: Alice or Bob is honest. Here, A_1, B_1 and A_2, B_2 represent Alice and Bob, respectively.

(1) If A_1 is honest and A_2 is dishonest, then:

$$\mathrm{REAL}_{\overline{A}}(W, Q) = \left\{F(W, A_2(Q)), A_2\left(\left(c_{1a}^i, c_{2a}^i\right), m_a, S\right)\right\}. \tag{10}$$

① Since A_1 is honest, B_1 sends a correct W to TTP, and the protocol will be executed correctly.

② What B_2 sends to TTP depends on the actual selection of A_2. B_2 sends Q to A_2 under the ideal model. A_2 sends $A_2(Q)$ to B_2 in the practical cases, and B_2 sends $A_2(Q)$ to TTP. Finally, TTP outputs $F(W, A_2(Q))$.

③ Ideally, B_2 uses the $F(W, A_2(Q))$ sent by TTP to try to get $\mathrm{view}_{B_2} F(W, A_2(Q))$ that is indistinguishable from the $\mathrm{view}_{A_2} F(W, A_2(Q))$ calculated by A_2 in practice and make it the output of A_2 in the practical cases.

That is, B_2 selects W' to make $F(A_1(W), Q\prime) = F(A_1(W), Q)$, performs all the calculations in Algorithm 2, obtains m_a' and $c_{1a}^{i'}, c_{2a}^{i'}$, and records the received sequence S' in the zero-knowledge proof. Thus, the protocol proceeds, and we get

$$\{\mathrm{IDEAL}_{\overline{B}}(W, Q)\} = \left\{F(W, A_2(Q)), A_2\left(\left(c_{1a}^{i'}, c_{2a}^{i'}\right), m_a', S'\right)\right\}. \tag{11}$$

Because ciphertexts are encrypted ideally and practically using the same probability algorithm, there are $c_{1a}^{i'}{}^c \equiv c_{1a}^i$ and $c_{2a}^{i'}{}^c \equiv c_{2a}^i$. The random number a_i is indistinguishable from a_i', so $\left\{\mathrm{REAL}_{\overline{A}}(W, Q)\right\} = \{\mathrm{IDEAL}_{\overline{B}}(W, Q)\}$.

(2) If A_1 is dishonest and A_2 is honest, there are two situations:

Actually, A_1 completes the zero-knowledge proof and publishes the results:

$$\mathrm{REAL}_{\overline{A}}(W, Q) = \left\{A_1\left(\left(c_{1b}^i, c_{2b}^i\right), m_b, S\right), F(W, Q)\right\}. \tag{12}$$

Actually, A_1 does not publish the results or execute the zero-knowledge proof:

$$\mathrm{REAL}_{\overline{A}}(W, Q) = \left\{A_1\left(\left(c_{1b}^i, c_{2b}^i\right), m_b, S\right), \perp\right\}. \tag{13}$$

① Because A_2 is honest, B_2 will send correct Q to TTP, and the protocol will be carried out correctly.

② What B_1 will send to TTP depends on the choice of A_1 in the practical situation. Ideally, B_1 sends W to A_1; practically, A_1 sends $A_1(W)$ to B_1, and B_1 sends $A_1(W)$ to TTP. Finally, TTP outputs $F(A_1(W), Q)$.

③ If A_1 does not publish the results or do not conduct the zero-knowledge proof in practice, B_2 in the ideal model will get $\perp$ from TTP.

④ Ideally, B_1 uses the $F(A_1(W), Q)$ sent by TTP to try to obtain $\mathrm{view}_{B_1}(A_1(W), Q)$ that is indistinguishable

from $\text{view}_{A_1}(A_1(W), Q)$ calculated by A_1 in the practical situation and to make it to be the output of A_1 in the practical situation.

That is, B_1 selects Q' to make $F(A_1(W), Q') = F(A_1(W), Q)$. And Algorithm 2 is carried out. Finally, under the ideal model, B_1 obtains m_b' and $c_{1b}^{i'}, c_{2b}^{i'}$ and records the sequence S' received through the zero-knowledge proof. In this way, we get the following.

Ideally, when B_1 does not publish results to B_2 via TTP:

$$\text{IDEAL}_{\overline{B}}(W, Q) = \left\{A_1\left(\left(c_{1b}^{i'}, c_{2b}^{i'}\right), m_b', S'\right), \perp\right\}. \tag{14}$$

Ideally, when B_1 announces results to B_2 via TTP:

$$\text{IDEAL}_{,\overline{B}}(W, Q) = \left\{A_1\left(\left(c_{1b}^{i'}, c_{2b}^{i'}\right), m_b', S'\right), F\left(A_1(W), Q\right)\right\}, \tag{15}$$

where $c_{1b}^{i'}, c_{2b}^{i'}$, and c_{1b}^{i}, c_{2b}^{i} are encrypted by the same ECC encryption algorithm. m_b' and m_b are computed by both random numbers and constant operations. The zero-knowledge proof guarantees $S'^c \equiv S$. Therefore, for any algorithm in the practical protocol $\overline{A} = (A_1, A_2)$, there exists $\overline{B} = (B_1, B_2)$ in the ideal protocol, which makes

$$\left\{\text{IDEAL}_{\overline{B}}(W, Q)\right\} \stackrel{c}{\equiv} \left\{\text{REAL}_{\overline{A}}(W, Q)\right\}. \tag{16}$$

Thus, the protocol's security is proven.

5. Efficiency Analysis

5.1. Computational Complexity. Reference [24] proposed a protocol to solve the millionaires' problem based on the decision Diffie–Hellman hypothesis (DDH) and performed $4nt + n$ modular multiplications. Reference [25] designed a millionaires' problem comparison protocol with 0-1 coding rules based on ElGamal encryption, which needs $(2m + 3)\log P + 5m$ modular multiplications. Reference [22] designed an antimalicious millionaires' problem protocol based on the Paillier encryption and performed $10m \log N + 2$ modular multiplications.

During the execution of Algorithm 1, the computational complexity mainly includes: m times ECC encryption operations of Alice and 1 time ECC decryption operation of Bob, with a total of $2m + 1$ modular multiplications. During the execution of Algorithm 2, the computational complexity mainly includes: m times ECC encryption operations and 1 time ECC decryption operation of Alice and several modular multiplication operations during message verification. A total of $11m + 10$ modular multiplications are performed, and the rest are ordinary multiplication and addition operations, which can be ignored.

5.2. Communication Complexity. There are two rounds of communication in Algorithm 1; reference [24] carried out three rounds of communication; reference [25] carried out three rounds of communication; reference [22] carried out three rounds of communication; and Algorithm 2 carried out six rounds of communication, as shown in Table 2.

5.3. Experimental Simulation. To verify the validity of the above protocols more intuitively, we compare Algorithm 2 with reference [22, 24, 25]. The experimental environment is Windows10 (64 bit) operating system, Intel (R) Core (TM) i7-5500U CPU @ 2.40 GHz processor, 8.00 GB memory, and the experiment is carried out by Python language.

The Paillier encryption, ElGamal encryption, and GM encryption have the same size of inputs in the experiment, and the time of protocol preprocessing is ignored in the experiment. Figure 2 is a comparison of the protocol time consumption in experiment 1 with the increase of modulus. The data held by each participant in experiment 1 is an integer from 0 to 100 (set length is set to 100). The average execution time (the ordinate coordinate) of the four protocols is calculated under 128, 256, 512, and 1,024 bit modules (the horizontal coordinate). As can be seen from Figure 2, the time consumed by Algorithm 2 is lower than those of other references.

Figure 3 is a comparison of the execution time of experiment 2 with the increase of data range. In experiment 2, each participant held the input data in the range of 0–100, 100–200, 200–300, 300–400, 400–500, 500–600, 600–700, 700–800, 800–900, and 900–1,000 under the same module. As can be seen from Figure 3, Algorithm 2 takes less time in different data ranges than other protocols, and the time consumption is relatively stable.

The results show that under the same security performance, Algorithm 2 is slightly more efficient than reference [24] and has obvious advantages than Reference [22, 25]. Algorithm 2 can resist malicious attacks and has higher security and greater practical value. (Note: For Algorithm 2, increased the bitcoin commitment, cut-choose, and zero-knowledge proof methods will result in significantly higher computational complexity and lower execution efficiency, making malicious model protocols no more efficient than semihonest model protocols. However, preprocessing or computing outsourcing can be used to improve efficiency, and both methods are available in Algorithm 2).

5.4. Applications. This paper uses efficient ECC encryption to design and study the classic millionaires' problem in SMC. It not only solves the problem of comparison between two numbers but also distinguishes whether two numbers are equal or not. The protocol can be widely applied to sort confidentially. Alice has $X = (a_1, a_2, ..., a_m)$; Bob has y; Bob wants to query the ranking position of y in X; and both sides do not want to expose any information about X and y. This problem is an important application of SMC in data query and has wide application prospects, such as secret ranking of college entrance examination results: after the college entrance examination, candidates want to check their ranking in the reported candidates, and the school does not want to disclose any information about other candidates. The problem can be solved by our protocol.

The same method can be applied to the smart contract quality evaluation in the blockchain. Blockchain technology is considered to be the next generation of disruptive core technology after steam engine, power, and Internet. In order

Table 2: Performance comparison of the efficiency of the four protocols in terms of computational complexity, number of communication rounds, and whether they can resist malicious attacks.

Protocol	Fair for both parties	Computational complexity (modular multiplications)	Communication (rounds)	Resist malicious attacks
Algorithm 1	No	$2m + 1$	2	×
Reference [24]	No	$4nt + n$	$3n \log P$	×
Reference [25]	No	$(2m + 3)\log P + 5m$	3	×
Reference [22]	Yes	$10m \log N + 2$	3	√
Algorithm 2	Yes	$11m + 10$	6	√

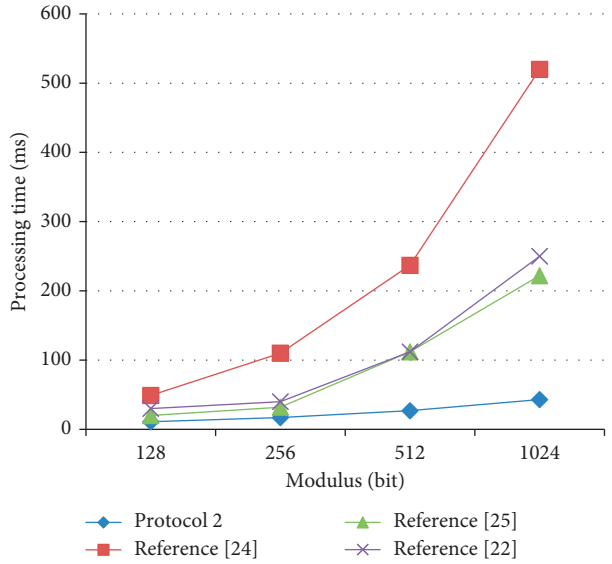

Figure 2: Time consumption comparison of different modules. The data held by each participant is an integer from 0 to 100 (set length is set to 100).

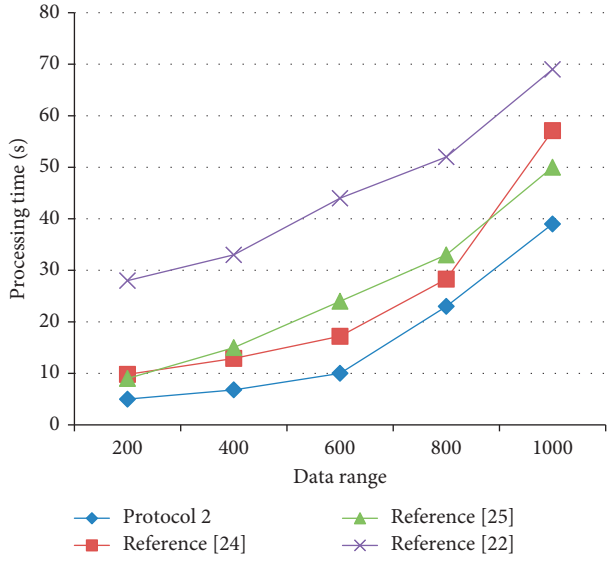

Figure 3: Time consumption comparison for different input ranges under the same module.

to establish a high-quality blockchain application environment, excellent smart contract developers will be rewarded by obtaining some form, such as tokens. However, there is no good evaluation method for the quality of the smart contract. Most of them are sorted by the number of contract calls and the total amount of contract transactions. Therefore, using our protocol to construct the sorting method, the screened high-ranking users have strong authenticity and security and are not easy to forge.

6. Summary and Prospect

As the cornerstone of SMC, the millionaires' problem is still under constant researches by experts and scholars. However, most of the schemes are designed under the semihonest model, which cannot resist malicious attacks and affect the practical application of the protocols. This paper uses the 0-1 encoding method and ECC encryption algorithm, first designs a semihonest model protocol, and then improves it for malicious behaviors. The millionaires' problem protocol under the malicious model solves the problem of malicious attacks in practical applications. By comparing with the existing protocols, our protocol is more efficient and practical.

Data Availability

The algorithm data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was supported by the fund project, National Natural Science Foundation of China (92046001 and 61962046); Inner Mongolia Natural Science Foundation (2021MS06006); Baotou Kundulun District Science and Technology Planning Project (YF2020013); Inner Mongolia Discipline Inspection And Supervision Big Data Laboratory open project fund (IMDBD2020020); Major Science And Technology Projects in Inner Mongolia (2019ZD025); basic scientific research business fee project of Beijing Municipal Commission of education (110052972027); the Scientific Research Launch Funds of North China University of Technology (110051360002); and Beijing Urban Governance Research Base of North China University of Technology.

References

[1] A. C. Yao, "Protocols for secure computations," in *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, pp. 160–164, IEEE press, Chicago, IL, USA, June 1982.

[2] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game," in *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, ACM*, pp. 218–229, Montreal, Canada, June 1987.

[3] O. Goldreich, *Foundations of Cryptography-Volume 2: Basic Applications*, Cambridge University Press, Cambridge, UK, 2009.

[4] T. Li, Z. Wang, G. Yang, and Y. Cui, "Semi-selfish mining based on hidden Markov decision process," *International Journal of Intelligent Systems*, vol. 36, 2021.

[5] Y. Chen, J. Sun, Y. Yang, and T. Li, "PSSPR: a source location privacy protection scheme based on sector phantom routing in WSNs," *International Journal of Intelligent Systems*, vol. 37, 2021.

[6] T. Wang, W. Ma, and W. Luo, "Information sharing and secure multi-party computing model based on blockchain," *Computer science*, vol. 46, no. 9, pp. 162–168, 2019.

[7] J. Liu, Y. Tian, Y. Zhou, Y. Xiao, and N. Ansari, "Privacy preserving distributed data mining based on secure multi-party computation," *Computer Communications*, vol. 153, pp. 208–216, 2020.

[8] G. Xu, Y. Cao, S. Xu et al., "A novel post-quantum blind signature for log system in blockchain," *Computer Systems Science and Engineering*, vol. 41, no. 3, pp. 945–958, 2022.

[9] M. Blatt, A. Gusev, Y. Polyakov, and S. Goldwasser, "Secure large-scale genome-wide association studies using homomorphic encryption," *Proceedings of the National Academy of Sciences*, vol. 117, no. 21, pp. 11608–11613, 2020.

[10] X. Liu, R. Zhang, G. Xu, X.-B. Chen, and N. N. Xiong, "Confidentially judging the relationship between an integer and an interval against malicious adversaries and its applications," *Computer Communications*, vol. 180, pp. 115–125, 2021.

[11] L. Feng, Y. Jie, and L. Zhibin, "A secure multi-party computation protocol for universal data privacy protection based on blockchain," *Journal of Computer Research and Development*, vol. 58, no. 2, p. 281, 2021.

[12] D. Li, X. Liao, T. Xiang, J. Wu, and J. Le, "Privacy-preserving self-serviced medical diagnosis scheme based on secure multi-party computation," *Computers & Security*, vol. 90, Article ID 101701, 2020.

[13] S. Parthasarathy, A. Harikrishnan, and G. Narayanan, "Secure distributed medical record storage using blockchain and emergency sharing using multi-party computation," in *Proceedings of the 2021 11th IFIP International Conference On New Technologies, Mobility And Security (NTMS)*, pp. 1–5, Paris, France, April 2021.

[14] W. Liu, Y.-B. Wang, A.-N. Sui, and M.-Y. Ma, "Quantum protocol for millionaire problem," *International Journal of Theoretical Physics*, vol. 58, no. 7, pp. 2106–2114, 2019.

[15] M. Hezaveh and C. Adams, "An efficient solution to the socialist millionaires' problem," in *Proceedings of the 2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 1–4, IEEE, Windsor, Canada, April 2017.

[16] S. Li and M. Zhang, "Efficient solutions to the problem of blind millionaires," *Journal of Computer Science*, vol. 43, no. 9, pp. 1755–1768, 2020.

[17] J. Zhang, H. Zheng, B. Ge, Y. Tang, and Q. Ye, "An efficient millionaire problem protocol and its application," *Computer Engineering*, vol. 47, no. 2, pp. 168–175, 2021.

[18] S. Ibrahim and A. Alharbi, "Efficient image encryption scheme using henon map, dynamic S-boxes and elliptic curve cryptography," *IEEE Access*, vol. 8, pp. 194289–194302, 2020.

[19] F. De Rango, G. Potrino, M. Tropea, and P. Fazio, "Energy-aware dynamic internet of things security system based on elliptic curve cryptography and message queue telemetry

transport protocol for mitigating replay attacks," *Pervasive and Mobile Computing*, vol. 61, Article ID 101105, 2020.

[20] R. Qazi, K. N. Qureshi, F. Bashir, N. U. Islam, S. Iqbal, and A. Arshad, "Security protocol using elliptic curve cryptography algorithm for wireless sensor networks," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 1, pp. 547–566, 2021.

[21] A. De Santis, S. Micali, and G. Persiano, "Non-interactive zero-knowledge proof systems," in *Proceedings of the Conference on the Theory and Application of Cryptographic Techniques*, pp. 52–72, Springer, Santa Barbara, CA, USA, August 1987.

[22] S. Li, W. Wang, and R. Du, "Solutions to the millionaire problem against malicious enemies," *Chinese Science: Information Science*, vol. 51, no. 1, pp. 75–88, 2021.

[23] B. Yang, *Modern Cryptography*pp. 124–128, Tsinghua University Press, Beijing, China, 4th edition, 2017.

[24] M. Liu, Y. Luo, P. Nanda, S. Yu, and J. Zhang, "Efficient solution to the millionaires'," *Computational Intelligence*, vol. 35, no. 3, pp. 555–576, 2019.

[25] Z. Li, L. Chen, Z. Chen, Y. Liu, and T. Gao, "An efficient millionaire problem protocol based on 1-r coding and its application," *Acta cryptographica Sinica*, vol. 6, no. 1, pp. 50–60, 2019.