electronics

MDPI

Article

Securely Computing Protocol of Set Intersection under the Malicious Model

Xin Liu [1,2], Weitong Chen [1], Neal Xiong [3], Dan Luo [4,*], Gang Xu [5] and Xiubo Chen [6]

[1] School of Information Engineering, Inner Mongolia University of Science and Technology, Baotou 014010, China; liuxin@nkddjx.wecom.work (X.L.); 2022023240@stu.imust.edu.cn (W.C.)
[2] School of Computer Science, Shaanxi Normal University, Xi'an 710062, China
[3] Department of Computer Science and Mathematics, Sul Ross State University, Alpine, TX 79830, USA; xiongnaixue@gmail.com
[4] Department of Computer, Tianjin Ren'ai College, Tianjin 301636, China
[5] College of Information, North China University of Technology, Beijing 100144, China; gx@ncut.edu.cn
[6] Information Security Center, State Key Laboratory of Network and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China; xb_chen@bupt.edu.cn
* Correspondence: rdjeans@gmail.com; Tel.: +86-13619961021

Abstract: Private set intersection (PSI) is a valuable technique with various practical applications, including secure matching of communication packets in the Internet of Things. However, most of the currently available two-party PSI protocols are based on the oblivious transfer (OT) protocol, which is computationally expensive and results in significant communication overhead. In this paper, we propose a new coding method to design a two-party PSI protocol under the semi-honest model. We analyze possible malicious attacks and then develop a PSI protocol under the malicious model using the Paillier cryptosystem, cut-and-choose, zero-knowledge proof, and other cryptographic tools. By adopting the real/ideal model paradigm, we prove the protocol's security under the malicious model, which is more efficient compared to the existing related schemes.

Keywords: secure multi-party computation; set intersection; malicious attacks; real/ideal model paradigm

Citation: Liu, X.; Chen, W.; Xiong, N.; Luo, D.; Xu, G.; Chen, X. Securely Computing Protocol of Set Intersection under the Malicious Model. *Electronics* **2023**, *12*, 2410. https://doi.org/10.3390/electronics12112410

Received: 30 April 2023
Revised: 19 May 2023
Accepted: 23 May 2023
Published: 26 May 2023

1. Introduction

With more and more attention paid to data value, secure data circulation will give full play to data value, which is conducive to accelerating social development. With the rapid development of blockchain, big data, and artificial intelligence, which are all major technological trends, joint computing from different data sources has great practical significance and has become the norm of computing. However, without effective prevention, data privacy and confidentiality are easily revealed in joint computing [1]. Therefore, protecting the confidentiality and privacy of data in joint computing is a serious challenge for network joint computing.

Secure multi-party computation (MPC), as the core technology of privacy computing, can protect data privacy and realize multi-party joint computing [2,3]. The Millionaires' problem proposed by Professor Yao Qizhi is the earliest MPC problem [4]. Goldreich, Cramer, and other researchers have further studied it [5,6], expanding the research field of MPC, including privacy preserving data mining, geometric computing, set problems, and confidential scientific computing [7,8]. Research has solved many practical problems in many fields and continuously promoted the development of MPC.

The secure computation of private set intersection (PSI) is a crucial research field in the realm of secure multiparty computation [9]. PSI has broad applications in various domains such as artificial intelligence, data mining, and the Internet of Things, including safeguarding privacy during data mining, discovering private address books, tracking COVID-19 close contacts, and more [10,11]. In the context of the Internet of Things, the

security module examines communication data packets between IoT devices and the network layer, and matches and filters them using the communication white list to enhance the security of network communication for IoT devices [12,13]. In the digital economy era, there is a high demand for both-sided PSI scenarios [14]. For example, in confidential social contact searches, finding common friends between two parties is also an application scenario for both-sided PSI.

An existing study [15] proposed a multi-party secure computation intersection scheme that can defend against malicious adversaries [16]; the authors in [17] proposed a privacy-preserving set intersection scheme based on Oblivious Linear Evaluation (OLE) primitives. Currently, most of the PSI schemes are built on Oblivious Transfer (OT) protocols [18], but OT-based protocols often require significant computational and storage costs and are only suitable for scenarios with large sets, and their advantages are not significant enough in small set scenarios [19].

The aim of this study is to introduce a highly efficient two-party secure multiparty computation (MPC) protocol designed for calculating set intersections under the malicious model. The key contributions of this paper are outlined below:

(1) This paper proposes an MPC protocol for set intersection under the semi-honest model, based on the Paillier encryption algorithm. It then analyzes potential malicious behaviors that could arise.
(2) Building on the semi-honest model protocol and considering potential malicious behaviors, this paper proposes an MPC protocol for computing set intersection under the malicious model, utilizing cryptographic tools such as cut-and-choose and zero-knowledge proof.
(3) Using the real/ideal model paradigm, the paper proves the security of the proposed protocol under the malicious model and analyzes and compares its efficiency.
(4) The efficacy of the proposed protocols is validated through a range of performance analyses, comparisons, and simulation experiments when compared to existing methodologies.

2. Related Work

As an important research field of MPC, the earliest PSI scheme [20] used the naive hash method to hash the set elements first, and then obtained the intersection by comparing the hash values. Although this scheme is very efficient, it is vulnerable to collision attack. In order to solve this problem, some PSI protocols use technologies such as efficient OT scheme, inadvertent pseudorandom function, and cuckoo hash [21,22], so as to achieve linear computational complexity and communication complexity.

Efraim et al. presented a PSI MPC protocol under the malicious model in study [15]. The authors in [15] introduce PSImple (Practical Multiparty Maliciously-Secure Private Set Intersection), which is the first maliciously secure multiparty PSI protocol that demonstrates concrete efficiency. The construction of PSImple is built upon the concepts of oblivious transfer and garbled Bloom filters. Furthermore, study [15] provides evidence that PSImple remains competitive with the most advanced concrete and efficient semi-honest multi-party PSI protocols available. However, the communication complexity of the protocol is considerable due to the use of garbled Bloom filter (GBF) [23], which requires extensive communication for its transmission.

In study [17], a novel method is introduced for calculating the intersection between sets using Oblivious Linear Function Evaluation (OLE) as a primitive. At an abstract level, the approach leverages OLE to effectively add two polynomials in a randomized manner while maintaining the roots of the resulting polynomials. By assigning the roots of the input polynomials to be the elements of the input sets, this directly leads to an intersection protocol with an optimal asymptotic communication complexity of $O(m\kappa)$. The protocol presented in study [17] achieves information-theoretic security against a malicious adversary under the assumption of OLE's availability. Nevertheless, this method exhibits high computational complexity.

Given the limitations of the aforementioned protocols, this paper proposes efficient MPC protocols for set intersection under both the semi-honest and malicious models, utilizing the Paillier encryption system, zero-knowledge proof, and cut-and-choose method. The protocol's efficiency is then compared with that of existing protocols to provide insight into its effectiveness.

3. Preliminary Knowledge

3.1. Paillier Cryptosystem

Paillier proposed an additive homomorphic cryptosystem in 1999 [24], which consists of three phases:

Key generation: Two large prime numbers p and q are randomly selected, and $N = pq$, with $\lambda = lcm(p-1, q-1)$ and $\gcd(pq, (p-1)(q-1)) = 1$. A random number $g \in Z^*_{N^2}$ is chosen as the public key, while λ is the private key.

Encryption: Let m be the message to be encrypted, where $0 \leq m < n$. Choose a random integer r where $0 < r < n$. Compute the ciphertext as:

$$c = g^m r^n \bmod n^2.$$

Decryption:

$$m = L(c^\lambda \bmod n^2)(L(g^\lambda \bmod n^2))^{-1} \bmod n.$$

Public key encryption, randomization, completeness, and resistance to attacks are the primary advantages of the Paillier cryptosystem. These advantages make the Paillier cryptosystem highly valuable for secure transmission and processing of encrypted data. The following provides a further explanation of these advantages:

(1) Public key encryption: The Paillier cryptosystem employs public and private keys for encryption and decryption operations. The public key can be publicly shared, allowing senders to encrypt data using the recipient's public key and transmitting it through insecure channels. Only the holder of the private key can decrypt the data, ensuring that even if the public key is compromised, attackers cannot access the plaintext. This public key encryption scheme ensures secure transmission.

(2) Randomization: In the Paillier cryptosystem, the encryption process involves the use of random numbers to encrypt the plaintext. Even if the same plaintext is encrypted multiple times, the resulting ciphertext will differ due to the utilization of different random numbers. This randomization feature enhances the security of the cryptosystem by preventing attackers from extracting information about the plaintext by analyzing multiple ciphertexts.

(3) Completeness: The Paillier cryptosystem is complete, enabling encryption and decryption of arbitrary integers without limitations on specific data ranges. This flexibility allows the cryptosystem to handle various types of data, regardless of their size, through encryption and decryption operations.

(4) Resistance to attacks: Extensive cryptographic research and analysis have demonstrated the security and reliability of the Paillier cryptosystem. It withstands many common cryptographic attacks, including chosen plaintext attacks, chosen ciphertext attacks, and active attacks. These attacks attempt to deduce information about the keys or plaintext by obtaining plaintext–ciphertext pairs or manipulating the ciphertext. However, the Paillier cryptosystem effectively thwarts these attacks.

In conclusion, the advantages of the Paillier cryptosystem can be made as an effective tool for secure transmission and processing of encrypted data.

3.2. Zero-Knowledge Proof

The concept of zero-knowledge proof, as introduced in [25,26], involves a prover convincing a verifier that they possess a particular knowledge or answer without divulging said knowledge or answer. A zero-knowledge proof scheme is deemed secure if it satisfies three properties:

(1) Completeness: If the statement being proven is true, the verifier will always accept it.
(2) Soundness: If the statement being proven is false, the verifier will always reject it, and the probability of the prover successfully persuading the verifier is negligible.
(3) Zero-knowledge: During the proof, the verifier does not learn any information about the knowledge or answer being proven by the prover.

Zero-knowledge proofs offer the following benefits:

(1) Privacy safeguarding: Zero-knowledge proofs enable a prover to validate a statement's veracity to a verifier without disclosing any specific details about the statement. By solely presenting the essential proof while keeping sensitive data concealed, individual privacy is upheld.
(2) Preservation of confidentiality: Zero-knowledge proofs ensure that the verifier cannot gain access to additional information regarding the statement, apart from verifying its validity. Although the prover can establish the truthfulness of a fact, the verifier cannot extract the specific information underpinning the proof, thereby maintaining confidentiality.
(3) Reliability and verifiability: Zero-knowledge proofs assure that the prover can correctly construct the proof in accordance with predefined rules and protocols, allowing the verifier to verify its validity. This enhances the proof's dependability and the overall system's verifiability.
(4) Efficiency: Zero-knowledge proofs can be executed within relatively short timeframes without excessive computational demands. This renders them practical and efficient for real-world applications.

To summarize, the advantages of zero-knowledge proofs encompass privacy protection, confidentiality preservation, reliability and verifiability, and efficiency. These advantages contribute to the high value of zero-knowledge proofs in domains such as secure protocols, identity authentication, and privacy protection, which explains their adoption in this manuscript.

3.3. Cut-and-Choose Method

The cut-and-choose technique is a commonly used cryptographic tool in the field of cryptography [27] and is crucial in the development of secure multiparty computation protocols. The fundamental concept is that one party creates multiple duplicates of garbled circuits in the protocol, while the other party requests that a subset of the circuits be opened for scrutiny. If the inspection is successful, the remaining circuits are computed and the ultimate output of the circuits is determined.

In the field of cryptography, the cut-and-choose technique is a widely used interactive zero-knowledge proof protocol employed for validating a participant's accurate computation while preserving the confidentiality of private information. This approach offers several notable benefits:

(1) Security: The cut-and-choose method ensures robust security measures. Participants engage in interactions where one participant (usually the prover) executes the computation and submits evidence, while the other participant (the verifier) randomly selects and verifies a subset of the evidence to establish the correctness of the computation. This mechanism effectively thwarts attacks such as cheating, tampering, and forgery.
(2) Zero-knowledge property: The cut-and-choose method exhibits the zero-knowledge property, enabling the prover to demonstrate the correctness of their computation to the verifier without disclosing any sensitive inputs or computational methods. The

verifier solely verifies that the computation's outcome is correct, without requiring knowledge of the computation's specifics.

(3) Resilience against attacks: This technique demonstrates robust resistance against various types of attacks. Even if the prover conducts erroneous computations or attempts to deceive by manipulating certain evidence elements, the verifier can effectively identify errors or cheating behaviors through random selection and verification of a portion of the evidence.

(4) Scalability: The cut-and-choose method can be readily scaled to accommodate different application scenarios and computational complexities. Enhancing the precision and reliability of verification can be achieved by increasing the number of evidence elements or selecting a larger number of random samples.

(5) Generality: The cut-and-choose method serves as a versatile zero-knowledge proof protocol applicable across diverse domains and computational tasks. It finds utility in verifying cryptographic protocols, password cracking results, data privacy, and numerous other applications.

Given these aforementioned advantages, the utilization of the cut-and-choose method in this manuscript enhances the protocol's security.

3.4. Security under the Malicious Model

The real versus ideal model paradigm [28] is a well-established technique for proving security in the context of malicious models. This approach guarantees that the actual protocol provides a comparable level of security as the ideal model.

During the computation process, both parties involved rely on a trusted third party (TTP) to carry out the computation. In the ideal scenario, both parties are only able to obtain their respective results and no other information. To prove the security of the protocol under the malicious model, the results obtained through computation in this model must be identical to those obtained in the ideal model. However, it should be noted that in the malicious model, the protocol must ensure that at least one party is honest during execution; otherwise, the protocol cannot be deemed secure and dependable. The security definition under the malicious model can be found in [29].

3.5. Ranking Method

The fundamental principle and detailed protocol for the ranking method, where identical numbers are assigned the same rank, are presented in [30]. Specifically, each identical data in the array are assigned the same ordinal bit, and the ordinal bit of the next larger data are increased by only one bit. This approach of sorting treats the same elements in multiple arrays as a single element, and the outcome is equivalent to sorting only the distinct data in the array.

Problem Description: There are n participants $P_1, \ldots, P_n$. For each $i \in [1, n]$, each P_i has a private and orderly array $T_i = (t_{i1}, \ldots, t_{ie_i})$; T_i is a standard array (no duplicate elements appear in the array). N participants need to conduct confidential calculation through cooperation. After the calculation, participants P_i can only get the sorting position r_{is} in the joint array $T = (T_1, \ldots, T_n)$ of each element t_{is}; $s \in [1, e_i]$ in their own array.

Calculation principle:

(1) Participants jointly agree on a complete set $J = [1, N]$, satisfying $T_i \subseteq J$. Under the complete set J, each participant P_i constructs an n-dimension vector according to their own array T_i:

$$Y_i = (y_{i1}, \ldots, y_{iN}) \tag{1}$$

where for each $j \in J$, define:

$$y_{ij} = \begin{cases} 1, & j \in T_i \\ 0, & j \notin T_i \end{cases} \tag{2}$$

(2) Let ${Y_1}^* = Y_1$, $P_i (i = 1, \ldots, n-1)$ take turns sending the vector ${Y_i}^*$ to P_{i+1}; P_{i+1} construct a new vector according to ${Y_i}^*$ and Y_{i+1}:

$$Y^*_{i+1} = (y^*_{(i+1)1}, \cdots, y^*_{(i+1)N})$$

where for each $j \in J$, define:

$$y^*_{(i+1)j} = \begin{cases} y_{(i+1)j}, & y_{(i+1)j} = 1 \\ y^*_{ij}, & y_{(i+1)j} = 0 \end{cases} \tag{3}$$

(3) P_i calculates the sorting position in the following way:

$$r_{is} = \sum_{j=1}^{t_{is}} y^*_{nj} \tag{4}$$

3.6. Coding Method

Coding method 1. Participants P_i $(i = 1, 2)$ construct the coding vector of their own sets through the ranking method in which the same numbers have the same order [30]: P_i convert their private sets S_i into an ordered array $T_i = (t_{i1}, \ldots, t_{ie_i})$, and participants P_i agree on a complete set $J = [1, N]$ satisfying $T_i \subseteq J$. The two participants use the ranking method for cooperative confidential calculation. After the calculation, the participants P_i could only know the sorting position r_{is} of each element t_{is}; $s \in [1, e_i]$ in their combined array $T = (T_1, T_2)$. At the same time, P_i can know the order ${r'}_{is}$ $(s = 1, \ldots, e_i)$ of each element in their set. Make the set $U = \{u_1, \ldots, u_n\}$ which contains N elements. The process in which the participants P_i encode the private set S_i as a vector V_i is as follows: P_i make the elements in which the order are r_{is} in the set U (i.e., $k = r_{is}$, where k is the subscript of element u in the set U) which be random even numbers, and the elements in other positions be random odd numbers, then P_i gets $V_i = (v_{i,1}, \ldots, v_{i,n})$, an n-dimension vector, respectively.

If the random odd or even numbers selected in the coding process are different, the coding vectors are different, that is, a set can be coded into many different vectors.

Example 1. *The following uses plaintext to demonstrate the coding process: Assume that participants P_1, P_2 have sets $S_1 = \{1, 3\}$ and $S_2 = \{3, 2, 6\}$, respectively. They convert S_1, S_2 to an ordered array $T_1 = (t_{11}, t_{12}) = (1, 3)$ and $T_2 = (t_{21}, t_{22}, t_{23}) = (2, 3, 6)$. Select $N = 7$, the complete set $U = \{u_1, \ldots, u_7\} = \{0, \ldots, 0\}$, and two participants construct 7-dimensional vectors $Y_1 = (1, 0, 1, 0, 0, 0, 0)$, $Y_2 = (0, 1, 1, 0, 0, 1, 0)$, respectively, according to Formula (1) in Section 3.5. P_1 sends the vector ${Y_1}^* = Y_1$ to P_2, P_2 gets $Y^*_2 = (1, 1, 1, 0, 0, 1, 0)$ according to the Formula (3). According to Y^*_2, each participant determines the sorting position of the elements of their respective array in the joint array according to Formula (4), and the sorting result is shown in Table 1.*

Table 1. Results of the ranking method in which the same numbers have the same order.

Element	Algorithm	Sorting Position
1	1	1
2	1 + 1	2
3	1 + 1 + 1	3
6	1 + 1 + 1 + 0 + 0 + 1	4

According to Table 1, P_i makes the elements in which order are r_{is} in the set U (i.e., $k = r_{is}$, where k is the subscript of element u in the set U) and be random even numbers, and the elements at other positions be random odd numbers, then P_1 gets an n-dimension vector $V_1 = (2, 1, 6, 5, 3, 7, 9)$ and P_2 gets an n-dimension vector $V_2 = (3, 2, 8, 4, 3, 1, 9)$.

3.7. Transformation Problem of Set Intersection

Problem description

Suppose there are two participants, Alice and Bob. Alice has a private set $A = \{ e_{a,1}, \ldots, e_{a,l_a} \}$ and Bob has a private set $B = \{e_{b,1}, \ldots, e_{b,l_b}\}$. They want to compute a set $T = A \cap B$ without disclosing any additional information. After the implementation of the protocol, the participants cannot obtain any additional information except the intersection T and inferable information from T.

Problem transformation

Alice encodes A to $V_a = (x_1, \ldots, x_n)$, Bob encodes B to $V_b = (y_1, \ldots, y_n)$, and makes the set $T = \varnothing$. For x_1 and $y_1, \ldots, x_n$, and y_n, Alice or Bob gets the corresponding $a(x_k - y_k)$ $(k = 1, \ldots, n)$ through calculation by using the Paillier cryptosystem, where a is a random odd number, and then each further determines whether y_k or x_k is an odd number or an even number according to what they own x_k or y_k, then makes $T \leftarrow T \cup \{u_k\}$ or $T \leftarrow T \cup \varnothing$, and finally obtains the intersection T of A and B. Taking x_1 and y_1 as an example, the specific execution process is as follows:

Alice obtains $a(x_1 - y_1)$ by calculations. Alice owns x_1, and makes the set $T = \varnothing$. ① In the case that x_1 is an even number, if $a(x_1 - y_1)$ is an even number (where a is a random odd number), according to the properties of odd and even numbers, Alice can conclude that $x_1 - y_1$ is an even number, since x_1 is even, then y_1 is even. Then Alice makes $T \leftarrow T \cup \{u_1\}$. If $a(x_1 - y_1)$ is an odd number, Alice can conclude that $x_1 - y_1$ is an odd number, then y_1 is odd. Then Alice makes $T \leftarrow T \cup \varnothing$. ② In the case that x_1 is an odd number, Alice makes $T \leftarrow T \cup \varnothing$.

For x_2 and $y_2, \ldots, x_n$, and y_n, Alice performs similar calculations and judgments as the above execution. Eventually, Alice gets the intersection T of A and B, and tells Bob T.

According to the elements u_k $(k = 1, \ldots, n)$ in the intersection T, the elements of the corresponding order position $r'_{is} = k$ (where $i = 1, 2$ and $s = 1, \ldots, e_i$) in Alice and Bob's respective private sets are the plain elements of the intersection set T.

Example 2. *Based on Example 1, it demonstrates the judgment process in plaintext: Let the private sets of Alice and Bob be* $A = S_1 = \{1, 3\}$ *and* $B = S_2 = \{3, 2, 6\}$*, respectively, and the corresponding encoding vector be:*

$$V_a = (x_1, \ldots, x_n) = V_1 = (2, 1, 6, 5, 3, 7, 9),$$

$$V_b = (y_1, \ldots, y_n) = V_2 = (3, 2, 8, 4, 3, 1, 9).$$

For $x_1 = 2$ and $y_1 = 3$, let Bob take random odd numbers $a = 3$. Alice obtained $a(x_1 - y_1) = (3 \times (2 - 3) = -3)$ by calculations. Let the set $T = \varnothing$, because Alice knows a is odd, and $a(x_1 - y_1) = -3$ is odd; according to the operational properties of odd and even numbers, Alice can conclude that $x_1 - y_1$ is odd (value is -1), and has $x_1 = 2$ which is an even number, then concludes that y_1 (value is 3) is odd, which makes $T \leftarrow T \cup \varnothing$.

For x_2 and $y_2, \ldots, x_n$, and y_n, Alice performs similar calculations and judgments as the above execution. Eventually, Alice gets the intersection $T = \{u_3\}$ of A and B, and tells Bob T.

According to Table 1, the element $\{3\}$ of the corresponding order position $r'_{is} = k = 3$ (where $i = 1, 2$ and $s = 1, \ldots, e_i$) in the sets A and B is the plain element of the intersection set T, so the intersection $T = \{3\}$.

4. The MPC Protocol of Set Intersection under the Semi-Honest Model

The outline of the protocol is presented in Algorithm 1 and the detailed procedures of the Private Set Intersection (PSI) under the semi-honest model are elucidated in Protocol 1. The development of the MPC protocol in the presence of a malicious model is anchored on

the semi-honest protocol and takes into account potential malicious conduct [29]. Hence, it is crucial to examine both the protocol and malicious behaviors in Protocol 1.

Algorithm 1. Computing the set intersection under the semi-honest model.

Input: A: Alice's input; B: Bob's input; pk (g, N): public key; sk (λ): Alice's private key; *Encode*: encode with the Coding method 1; E: encrypt by the Paillier system; c: ciphertexts; D: decrypt the ciphertexts.
(1) $Encode(A) = V_a = (x_1, \ldots, x_n)$
(2) $Encode(B) = V_b = (y_1, \ldots, y_n)$
(3) $E_{pk}(x) = c_1 = g^{x_1} r_1^N \bmod N^2$
(4) Select a and r_2
(5) Compute $c_2 = c_1^a (g^{ay_1})^{-1} r_2^N \bmod N^2 = g^{a(x_1 - y_1)} (r_1^a r_2)^N \bmod N^2$
(6) $D(c_2) = a(x_1 - y_1)$
(7) Make $T = \varnothing$
(8) If x_1 is even and $a(x_1 - y_1)$ is even then
$x_1 - y_1$ is even;
y_1 is even;
make $T \leftarrow T \cup \{u_1\}$;
else if $a(x_1 - y_1)$ is odd then
$x_1 - y_1$ is odd;
y_1 is odd;
make $T \leftarrow T \cup \varnothing$;
else if x_1 is odd then
make $T \leftarrow T \cup \varnothing$
(9) Perform the steps (3)–(8) for x_2 and y_2,. . ., x_n and y_n
(10) Obtain the intersection T
(11) Obtain plain elements of T according to the elements u_k $(k = 1, \ldots, n)$ in the T
Output: Intersection of A and B.

Protocol 1. The MPC protocol of set intersection under the semi-honest model.

Input: Private set A of Alice, private set B of Bob.
Output: The intersection T of A and B.
Preparation: The Paillier cryptosystem's public and private keys, (g, N) and λ respectively, are created by Alice. Following this, Alice transmits the public key to Bob.
(1) Alice encodes A to $V_a = (x_1, \ldots, x_n)$ and Bob encodes B to $V_b = (y_1, \ldots, y_n)$. Since the following calculations are the same for x_1 and $y_1, \ldots, x_n$, and y_n, the calculation of x_1 and y_1 are described as an example. In each of the following steps, x_2 and $y_2, \ldots, x_n$, and y_n are calculated at the same time as x_1 and y_1.
(2) Alice encrypts x_1 with the public key to $c_1 = g^{x_1} r_1^N \bmod N^2$ and sends c_1 to Bob.
(3) Then Bob selects new random numbers a (a takes a random odd number) and r_2 to calculate $c_2 = c_1^a (g^{ay_1})^{-1} r_2^N \bmod N^2 = g^{a(x_1 - y_1)} (r_1^a r_2)^N \bmod N^2$,
and sends c_2 to Alice.
(4) Alice gets $a(x_1 - y_1)$ by decrypting c_2. Alice owns x_1, and makes the set $T = \varnothing$. ① In the case that x_1 is an even number, if $a(x_1 - y_1)$ is an even number (where a is a random odd number), according to the properties of odd and even numbers, Alice can conclude that $x_1 - y_1$ is an even number, since x_1 is even, then y_1 is even. Then Alice makes $T \leftarrow T \cup \{u_1\}$. If $a(x_1 - y_1)$ is an odd number, Alice can conclude that $x_1 - y_1$ is an odd number, then y_1 is odd. Then Alice makes $T \leftarrow T \cup \varnothing$. ② In the case that x_1 is an odd number, Alice makes $T \leftarrow T \cup \varnothing$.
(5) For x_2 and y_2,. . ., x_n, and y_n, Alice and Bob perform the calculation and judgment in steps (2)–(4) above at the same time. Eventually, Alice obtains the intersection T of A and B.
(6) Alice sends T to Bob. According to the elements u_k $(k = 1, \ldots, n)$ in the intersection T, the elements of the corresponding order position $r'_{is} = k$ (where $i = 1, 2$ and $s = 1, \ldots, e_i$) in Alice and Bob's respective private sets are the plain elements of the intersection set T.
The protocol ends.

For Protocol 1 to be secure, it is imperative that both Alice and Bob adhere to the semi-honest model. Failure to do so would compromise the protocol's security, and as such, measures need to be taken to address any malicious behavior in the semi-honest model protocol. This ensures the protocol can withstand potential attacks by malicious participants.

5. The MPC Protocol of Set Intersection under the Malicious Model

Solution: A widely used approach to designing MPC protocols for the malicious model is to examine potential attacks on protocols in the semi-honest model and devise corresponding countermeasures [29]. This ensures that malicious parties cannot engage in harmful activities, or if they do, they can be detected by the other party.

It is worth noting that certain malicious behaviors that cannot be prevented under the ideal model are similarly unavoidable under the malicious model. Nonetheless, protocols under the malicious model must still maintain the same level of security as those under the ideal model. Malicious behaviors such as refusal to participate, false input, and premature termination of protocols are not taken into consideration in actual protocols.

In Protocol 1, possible malicious acts include:

(1) If Bob is characterized as being semi-honest, Alice may engage in deceitful behavior, such as deliberately conveying an incorrect set T to Bob at the conclusion of the protocol, resulting in an inaccurate outcome for Bob. It is inequitable for either party to disclose the computation results to the other.

(2) When Alice is semi-honest, Bob can perform malicious acts such as: ① Bob does not choose a real random number r_2 when calculating $c_2 = g^{a(x_i - y_i)}({r_1}^a r_2)^N \bmod N^2$. However, as the decryption has eliminated the impact of r_2, Bob cannot get any private information from r_2. ② If a selected by Bob is not a random odd number, but a random even number, then when Alice publishes the value of $a(x_k - y_k)$, Bob will get the correct result, but Alice will not get the correct result.

In view of the above malicious acts, Protocol 1 is improved by using such cryptographic methods such as cut-and-choose method and zero-knowledge proof, and Alice and Bob jointly generate random odd numbers a in Protocol 1. Neither Alice nor Bob knows the true value of a, but they can use a to complete the above protocol.

5.1. Specific Protocol

It is important to acknowledge that while the protocol cannot eliminate the possibility of participants engaging in malicious behavior, it is capable of detecting such behavior. During the execution of the protocol, if Alice is truthful while Bob is being deceptive, Alice can identify Bob's malicious actions. Conversely, Bob can also detect any deceptive behavior on Alice's part. However, if both Alice and Bob are acting maliciously, it has been demonstrated in theory that designing an MPC protocol that can address this scenario is impossible, and thus will not be further discussed. The outline of the protocol is presented in Algorithm 2, and the step-by-step procedure for the protocol is presented in Protocol 2.

5.2. Correctness Analysis

Protocol 2 treats Alice and Bob as equal, therefore only Alice's implementation is subject to analysis.

(1) Step (1) in Protocol 2 is for the participants to obtain the coding vectors of their own sets. In this process, Alice converts the elements in her set into random even numbers in the vector V_a, and the elements that do not exist into random odd numbers. In this way, it is avoided to directly use the original data for calculation.

(2) In step (2), Alice publishes $\left(c_{1a}^i, c_{2a}^i\right)$ $(i = 1, \ldots, m)$, but the published information is encrypted, and Bob cannot obtain any valuable information.

(3) Steps (3)–(5) employ the cut-and-choose technique to ascertain the presence of any malicious behavior among the participants.

Algorithm 2. Computing the set intersection under the malicious model.

Input: A: Alice's input; B: Bob's input; (g_a, N_a): public key generated by Alice; (g_b, N_b): Public key generated by Bob; λ_a: Alice's private key; λ_b: Bob's private key; *Encode*: encode with the Coding method 1; c: ciphertexts.

(1) Compute $u = g_a^{\lambda_a} \bmod N_a^2$

(2) Compute $v = g_b^{\lambda_b} \bmod N_b^2$

(3) Exchange (g_a, N_a, u) and (g_b, N_b, v)

(4) $Encode(A) = V_a = (x_1, \ldots, x_n)$

(5) $Encode(B) = V_b = (y_1, \ldots, y_n)$

(6) Select m odd numbers a_i $(i = 1, \ldots, m)$

(7) Select m odd numbers b_i $(i = 1, \ldots, m)$

(8) $\left(c_{1a}^i , c_{2a}^i\right) = \left(g_a^{a_i x_1} \bmod N_a^2 , g_a^{a_i} \bmod N_a^2\right)$

(9) $\left(c_{1b}^i , c_{2b}^i\right) = \left(g_b^{b_i y_1} \bmod N_b^2 , g_b^{b_i} \bmod N_b^2\right)$

(10) Select $m/2$ sets of $\left(c_{1b}^i , c_{2b}^i\right)$ from m sets of $\left(c_{1b}^i , c_{2b}^i\right)$

(11) Verify

If $b_i \bmod 2 \neq 0$ and $g_b^{b_i y_1} \bmod N_b^2 = c_{1b}^i$ then continue

else terminate

(12) Select $m/2$ sets of $\left(c_{1a}^i , c_{2a}^i\right)$ from m sets of $\left(c_{1a}^i , c_{2a}^i\right)$

(13) Verify

If $a_i \bmod 2 \neq 0$ and $g_a^{a_i x_1} \bmod N_a^2 = c_{1a}^i$ then continue

else terminate

(14) Select a group of $\left(c_{1b}^j , c_{2b}^j\right)$ and $\left(c_{1a}^i , c_{2a}^i\right)$ from the remaining $\left(c_{1b}^i , c_{2b}^i\right)$ and $\left(c_{1a}^i , c_{2a}^i\right)$

(15) Select random numbers $a \in Z_b^*$ and $b \in Z_a^*$

(16) $c_b = E_b\left(ab_j(x_1 - y_1)\right) = \left(c_{2b}^j\right)^{ax_1} \left(c_{1b}^j\right)^{-a} r_1^{N_b} \bmod N_b^2 = g_b^{ab_j(x_1 - y_1)} r_1^{N_b} \bmod N_b^2$

(17) $c_a = E_a(a_i b(x_1 - y_1)) = \left(c_{1a}^i\right)^b \left(c_{2a}^i\right)^{-by_1} r_2^{N_a} \bmod N_a^2 = g_a^{a_i b(x_1 - y_1)} r_2^{N_a} \bmod N_a^2$

(18) $m_a = c_a^{\lambda_a} \bmod N_a^2$

(19) $m_b = c_b^{\lambda_b} \bmod N_b^2$

(20) Exchange m_a and m_b

(21) Verify

If $\log_{c_a} m_a = \log_{g_a} u$ and $\log_{c_b} m_b = \log_{g_b} v$ then continue

else terminate

(22) Compute $L(m_a)/L(u)$ to obtain $a_i(x_1 - y_1)$

(23) Make $T = \varnothing$

(24) If y_1 is even and $a_i(x_1 - y_1)$ is even then

$x_1 - y_1$ is even;

x_1 is even;

make $T_b \leftarrow T_b \cup \{u_1\}$;

else if $a_i(x_1 - y_1)$ is odd then

$x_1 - y_1$ is odd;

x_1 is odd;

make $T_b \leftarrow T_b \cup \varnothing$;

else if y_1 is odd then

make $T_b \leftarrow T_b \cup \varnothing$

(25) Compute $L(m_b)/L(v)$ to obtain $b_j(x_1 - y_1)$

(26) Perform steps similar to step (24) to make $T_a \leftarrow T_a \cup \{u_1\}$ or $T_a \leftarrow T_a \cup \varnothing$

(27) Perform the steps (6)–(26) above for x_2 and y_2,..., x_n, and y_n

(28) Obtain T_a and T_b

(29) Obtain plain elements of T_a and T_b according to the elements u_k $(k = 1, \ldots, n)$ in the T_a and T_b

Output: Intersection of A and B.

Protocol 2. The MPC protocol of the set intersection under the malicious model.

Input: Private set A of Alice; private set B of Bob.
Output: The intersection T of A and B.
Preparation: Alice and Bob each create a public key, (g_a, N_a) and (g_b, N_b), respectively, for the Paillier cryptosystem, and compute values $u = g_a^{\lambda_a} \bmod N_a^2$ and $v = g_b^{\lambda_b} \bmod N_b^2$. They then exchange values (g_a, N_a, u) and (g_b, N_b, v).
(1) Alice encodes A to $V_a = (x_1, \ldots, x_n)$ and Bob encodes B to $V_b = (y_1, \ldots, y_n)$. Since the following calculations are the same for x_1 and y_1, …, x_n, and y_n, the calculation of x_1 and y_1 are described as an example. In each of the following steps, x_2 and y_2, …, x_n, and y_n are calculated at the same time as x_1 and y_1.
(2) For x_1 and y_1, Alice and Bob each randomly select m odd numbers, denoted as a_i and b_i, respectively, where $i = 1, \ldots, m$, and then compute the value of:
$\left(c_{1a}^i , c_{2a}^i\right) = \left(g_a^{a_i x_1} \bmod N_a^2 , g_a^{a_i} \bmod N_a^2\right)$, $\left(c_{1b}^i , c_{2b}^i\right) = \left(g_b^{b_i y_1} \bmod N_b^2 , g_b^{b_i} \bmod N_b^2\right)$,
then publish $\left(c_{1a}^i , c_{2a}^i\right)$, $\left(c_{1b}^i , c_{2b}^i\right)$, respectively.
(3) Alice employs the cut-and-choose technique to randomly select $m/2$ sets of $\left(c_{1b}^i , c_{2b}^i\right)$ from a total of m sets of $\left(c_{1b}^i , c_{2b}^i\right)$. She then requests Bob to make public the corresponding values of b_i and $g_b^{b_i y_1}$, which she subsequently validates using $b_i \bmod 2 \neq 0$ and $g_b^{b_i y_1} \bmod N_b^2 = c_{1b}^i$. The protocol proceeds if the verification is successful, otherwise it halts.
(4) Bob randomly chooses $m/2$ groups of $\left(c_{1a}^i , c_{2a}^i\right)$ from a set of m groups of $\left(c_{1a}^i , c_{2a}^i\right)$, and requests Alice to disclose the corresponding a_i and $g_a^{a_i x_1}$. Bob then verifies $a_i \bmod 2 \neq 0$ and $g_a^{a_i x_1} \bmod N_a^2 = c_{1a}^i$. If the verification is successful, Bob proceeds with the protocol; otherwise, the protocol terminates.
(5) Alice and Bob respectively and randomly select a group of $\left(c_{1b}^j , c_{2b}^j\right)$ and $\left(c_{1a}^i , c_{2a}^i\right)$ from the remaining $\left(c_{1b}^i , c_{2b}^i\right)$ and $\left(c_{1a}^i , c_{2a}^i\right)$, and respectively select random numbers $a \in Z_b^*$ and $b \in Z_a^*$.
Alice calculates:
$c_b = E_b\left(ab_j(x_1 - y_1)\right) = \left(c_{2b}^j\right)^{ax_1} \left(c_{1b}^j\right)^{-a} r_1^{N_b} \bmod N_b^2 = g_b^{ab_j(x_1 - y_1)} r_1^{N_b} \bmod N_b^2$,
Bob calculates:
$c_a = E_a(a_i b(x_1 - y_1)) = \left(c_{1a}^i\right)^b \left(c_{2a}^i\right)^{-by_1} r_2^{N_a} \bmod N_a^2 = g_a^{a_i b(x_1 - y_1)} r_2^{N_a} \bmod N_a^2$,
and they send the results to each other.
(6) Alice calculates $m_a = c_a^{\lambda_a} \bmod N_a^2$, Bob calculates $m_b = c_b^{\lambda_b} \bmod N_b^2$. Next, m_a and m_b exchange one another.
(7) In Section 3.2, both parties employ the zero-knowledge proof technique to demonstrate the accuracy of their computation results, thereby verifying $\log_{c_a} m_a = \log_{g_a} u$ and $\log_{c_b} m_b = \log_{g_b} v$. If one party fails to pass, it is proved to be malicious.
(8) If all are proved, Bob has the ability to compute $L(m_a)/L(u)$, which enables him to derive $a_i b(x_1 - y_1)$, and subsequently acquire $a_i(x_1 - y_1)$. Bob owns y_1, and makes the set $T_b = \varnothing$. ① In the case that y_1 is an even number, if $a_i(x_1 - y_1)$ is an even number, according to the properties of odd and even numbers, Bob can conclude that $x_1 - y_1$ is an even number, so x_1 is even. Then Bob makes $T_b \leftarrow T_b \cup \{u_1\}$. If $a_i(x_1 - y_1)$ is an odd number, Bob can conclude that $x_1 - y_1$ is an odd number, then x_1 is odd. Then Bob makes $T_b \leftarrow T_b \cup \varnothing$. ② In the case that y_1 is an odd number, Bob makes $T_b \leftarrow T_b \cup \varnothing$. Alice can calculate $L(m_b)/L(v)$ to obtain $ab_j(x_1 - y_1)$, and then obtain $b_j(x_1 - y_1)$. Alice uses similar methods above to make the set $T_a \leftarrow T_a \cup \{u_1\}$ or $T_a \leftarrow T_a \cup \varnothing$.
(9) Alice obtains the set T_a, Bob obtains the set T_b; the intersection of A and B, $T = T_a = T_b$. According to the elements u_k $(k = 1, \ldots, n)$ in the intersection T, the elements of the corresponding order position $r'_{is} = k$ (where $i = 1, 2$ and $s = 1, \ldots, e_i$) in Alice and Bob's respective private sets are the plain elements of the intersection set T.
The protocol ends.

(4) In step (7), Alice is required to prove the correctness of the decryption outcome m_a via a zero-knowledge proof. If the a_i in the remaining $m/2$ groups of $\left(c_{1a}^i , c_{2a}^i\right)$ are also random odd numbers, Bob can calculate $a_i(x_k - y_k)$ $(k = 1, \ldots, n)$ and judge the parity of x_k after publishing m_a.

(5) During the implementation of the protocol, the malicious act Alice may successfully perform is that Bob chooses a certain a_i which does not meet the requirements, and does not find it during the verification in step (3). In step (5), Bob mistakenly selects it, leading to an incorrect conclusion. Nevertheless, the information of y_k remains inaccessible to Alice, because $ab_j(x_k - y_k)$ is unsolvable for her (there are two unknowns in an equation), so Alice cannot judge the parity of y_k from it.

Alice employs the aforementioned techniques for dishonest purposes. The circumstances that offer the highest likelihood of success are as follows: among m groups of $\left(c_{1a}^i, c_{2a}^i\right)$ controlled by Alice, $m-1$ groups satisfy the stipulated criteria while only one group fails to do so, thereby yielding a maximum probability of success denoted as $1/m$. Assuming that $m = 20$, the most probable chance of successful deception amounts to $\frac{C_{19}^{10}}{C_{20}^{10}} \times \frac{1}{10} = \frac{1}{20}$. Conversely, if 10 groups do not meet the prescribed criteria, the probability of successful deception becomes $\frac{C_{10}^{10}}{C_{20}^{10}} = \frac{1}{184756}$. When $m = 40$, the probabilities are reduced to 2.5×10^{-2} and 7.3×10^{-12}, respectively. If more than $1/2$ groups fail to meet the requirements, the probability of successful deception dwindles to zero since it will be discovered during the verification phase. Consequently, the protocol is deemed secure.

(6) In steps (7)–(9), the two parties exchanged ciphertext and decrypted it by themselves, avoiding the situation that one party informed the other of the result, which is fair.

5.3. Security Proof

The security of Protocol 2 against malicious attacks is demonstrated through the application of the real/ideal model paradigm. This proof methodology follows the subsequent steps.

Theorem 1. *Protocol 2, expressed as* Π*, is established to be secure in the presence of malicious adversaries*.

Proof. For Protocol 2 to securely compute function F, the participants must identify the approved policy $\overline{A} = (A_1, A_2)$ in the actual protocol. This policy should be indistinguishable from the policy $\overline{B} = (B_1, B_2)$ used in the protocol under the ideal model. The security of the protocol can be demonstrated by establishing the indistinguishability between $\overline{A} = (A_1, A_2)$ and $\overline{B} = (B_1, B_2)$. □

In the protocol, at least one of A_1 and A_2 is honest, so there are two cases.

Case 1: A_1 is honest, A_2 is dishonest.

If A_1 is honest, executing the protocol Π will result in:

$$REAL_{\Pi,A}(x_k, y_k) = \{F(x_k, A_2(y_k), A_2(c_{1a}^i, c_{2a}^i), m_a, S\}(k = 1, \ldots, n)$$

The sequence of messages received by A_2 for zero-knowledge proof can be represented by the variable S.

Assuming A_1 is an honest participant who follows the protocol honestly, the behavior of B_1 can be determined. The objective is to demonstrate that A_2 in the actual protocol is indistinguishable from B_2 in the ideal model. To achieve this, it is necessary to identify an output strategy $\overline{B} = (B_1, B_2)$ under the ideal model that is indistinguishable from $REAL_{\Pi,\overline{A}(x_k,y_k)}$ in the actual model. When the protocol is executed, A_2 is the actual executor, and the correctness of the protocol must be confirmed based on the behavior $A_2(y_k)$ of A_2.

(1) During the execution of the protocol, since A_1 is an honest participant, it can be inferred that B_1 is also honest, and will replicate the actions of A_1 in transmitting the authentic information x_k to the trusted third party (TTP).

(2) During the execution of the protocol, as A_2 is acting in a dishonest manner, B_2 is also acting dishonestly. The information that they send to TTP is dependent on the policy of B_2, and the policy of B_2 aligns with the policy of A_2. Consequently, the input message that B_2 transmits to TTP is $A_2(y_2)$.

(3) The input information obtained by TTP is $(x_k, A_2(y_k))$, and $F(x_k, A_2(y_k))$ is calculated.

(4) B_2 gets $F(x_k, A_2(y_k))$ from TTP, uses $F(x_k, A_2(y_k))$ to get an $view_{B_2}^{F}(x_k, A_2(y_k))$ which is computational, indistinguishable from the $view_{A_2}^{\prod}(x_k, A_2(y_k))$ obtained by A_2 when the protocol is actually implemented, and hands $view_{B_2}^{F}(x_k, A_2(y_k))$ to A_2 to get the output of A_2. Subsequently, based on its own input and the protocol outcome, the simulator B_2 presumes that the input value of the other party satisfies the outcome, and conducts the protocol execution. In this case, B_2 selects x_k' to simulate the protocol and generates $F(x_k', A_2(y_k)) = F(x_k, A_2(y_k))$. The particular steps involved in the implementation process of B_2 are as follows:

① B_2 sends the information required in step (2) to A_2;
② After A_2 publishes the information in step (3), B_2 will verify it;
③ In step (4), B_2 will publish the information that A_2 requires A_1 to publish;
④ In step (5), B_2 chooses the necessary information from the remaining sets, computes the information, and then publicly discloses it;
⑤ Calculate m_a' in step (6) and publish it;
⑥ During step (7), zero-knowledge proof is utilized to authenticate the information, and as a result, B_2 acquires the message sequence S'.

B_2 obtains the following information during the execution of the protocol:

$$IDEAL_{F,B}(x_k, y_k) = \{F(x_k, A_2(y_k)), A_2(C_{1a}^{i'}, C_{2a}^{i'}), m_a', S'\}.$$

In steps (2)–(7) of the protocol, since the probability encryption algorithm adopted by the protocol is the same, then $(C_{1a}^{i}, C_{2a}^{i}) \stackrel{c}{\equiv} (C_{1a}^{i'}, C_{2a}^{i'})$, $m_a' \stackrel{c}{\equiv} m_a$. Zero-knowledge proof guarantees $S' \stackrel{c}{\equiv} S$. Then:

$$\{IDEAL_{F,B}(x_k, y_k)\} \stackrel{c}{\equiv} \{REAL_{\Pi,A}(x_k, y_k)\}.$$

Case 2: If A_1 is dishonest and A_2 is honest, there are two cases:

(1) If Alice ignores TTP after getting the information, TTP will send $\perp$ to Bob. Then:

$$REAL_{\Pi,A}(x_k, y_k) = \{A_1(c_{1b}^{i}, c_{2b}^{i}), m_b, S, \perp\}.$$

(2) Otherwise, TTP will send $F(A_1(x_k), y_k)$ to Bob, then:

$$REAL_{\Pi,A}(x_k, y_k) = \{A_1(c_{1b}^{i}, c_{2b}^{i}), m_b, S, F(A_1(x_k), y_k)\}(k = 1, \ldots, n).$$

The sequence of messages received by A_1 for zero-knowledge proof can be represented by the variable S.

As A_2 is an honest participant and executes the protocol as prescribed, the behavior of B_2 can be determined. The objective is to prove that A_1 in the actual protocol is indistinguishable from B_1 in the ideal model. To accomplish this, it is necessary to identify a policy $\overline{B} = (B_1, B_2)$ under the ideal model, whose output is indistinguishable from $REAL_{\Pi,A(x_k,y_k)}$ in the actual model. During the execution of the protocol, the actual executor is A_1. Hence, while proving the protocol's correctness, it is crucial to verify it based on the behavior $A_1(x_k)$ of A_1.

(1) During the execution of the actual protocol, A_1 is acting dishonestly; as a result, B_1 is also acting dishonestly. The information transmitted by B_1 to TTP is dependent on the policy of B_1, which is the same as the policy of A_1. Then B_1 will send $A_1(x_k)$ to TTP.
(2) During the execution of the actual protocol, as A_2 is an honest participant, it can be inferred that B_2 is also honest, and it sends TTP the real input information y_k.
(3) The input information obtained by TTP is $(A_1(x_k), y_k)$, and TTP calculates $F(A_1(x_k), y_k)$.
(4) B_1 uses $F(A_1(x_k), y_k)$ obtained from TTP to obtain $view^F_{B_1}(A_1(x_k), y_k)$, and $view^F_{B_1}(A_1(x_k), y_k)$ should be computationally indistinguishable from $view^{\prod}_{A_1}(A_1(x_k), y_k)$ obtained from the actual protocol implemented by A_1. The output of A_1 is obtained by handing over the execution of the protocol to A_1 after $view^F_{B_1}(A_1(x_k), y_k)$. Subsequently, B_1 conducts the protocol execution by presuming that the other party's input satisfies the outcome based on its own input and calculation results, that is, B_1 selects y'_k to simulate the protocol and makes $F(A_1(x_k), y'_k) = F(A_1(x_k), y'_k)$. The specific implementation process of B_1 is as follows:
    ① B_1 sends the information required in step (2) to A_1;
    ② In step (3), B_1 will publish the information that A_1 requires A_2 to publish;
    ③ After A_1 publishes the information in step (4) of the protocol, B_1 will verify it;
    ④ In step (5), B_1 chooses the necessary information from the remaining sets, computes the information, and then publicly discloses it;
    ⑤ Calculate m'_b in step (6) and publish it;
    ⑥ In step (7), ZKP is used to verify the information and B_1 obtains the message sequence S'.

When B_1 executes the protocol, there are two situations:

(1) If A_1 ignores TTP after getting the information, then:

$$IDEAL_{F,B}(x_k, y_k) = \{A_1(c^{i'}_{1b}, c^{i'}_{2b}), m'_b, S', \perp\}.$$

(2) Otherwise,

$$IDEAL_{F,B}(x_k, y_k) = \{A_1(c^{i'}_{1b}, c^{i'}_{2b}), m'_b, S', F(A_1(x_k), y'_k)\}.$$

In steps (2)–(7) of the protocol, since the probability encryption algorithm adopted by the protocol is the same, then $(C^i_{1b}, C^i_{2b}) \stackrel{c}{\equiv} (C^{i'}_{1b}, C^{i'}_{2b})$, $m'_b \stackrel{c}{\equiv} m_b$. Zero-knowledge proof guarantees $S' \stackrel{c}{\equiv} S$. Then:

$$\{REAL_{\Pi,A}(x_k, y_k)\} \stackrel{c}{\equiv} \{IDEAL_{F,B}(x_k, y_k)\}.$$

In conclusion, for any probabilistic polynomial-time policy $\overline{A} = (A_1, A_2)$ that is acceptable during the actual protocol execution, there exists a probabilistic polynomial-time policy $\overline{B} = (B_1, B_2)$ that is acceptable under the ideal model. This makes $IDEAL_{F,B}(x_k, y_k)$ and $REAL_{\Pi,A}(x_k, y_k)$ indistinguishable during computation. Consequently, Protocol 2 is secure under the malicious model.

6. Performance and Comparison of Protocols

The authors of [15,17] have implemented the calculation of PSI under the malicious model. In this study, the performance of Protocol 2 is evaluated by comparing its computational complexity, communication complexity, and experimental simulation time.

6.1. Computation Complexity

The protocol in study [15] is designed based on OT extension, and the computational complexity is $O(n^2)$. The protocol in study [17] is based on the objective linear function eval-

uation (OLE), so it requires a lot of public key encryption operations, and the computation complexity is $O(n\ lb^2\ n)$.

In this paper's Protocol 2, both Alice and Bob are required to encode their respective private sets into vectors using the secure ranking method where identical elements have the same order. The computational complexity of encoding vectors generated by both parties is denoted by $O(n)$. Additionally, Alice and Bob need to generate mn groups of modular index, where m is the number of randomly selected odd numbers a_i or b_i, and n is the number of elements in the set coding vector. Each party needs $2mn$ modular index operations, which require $4mn$ modular index operations in total. Both parties need to verify $n(m/2)$ modulus index and perform $n(m/2)$ modulus index operations, requiring a total of mn modulus index calculations. For each zero-knowledge proof of discrete logarithms, six modular exponential operations are performed, and each participant performs the zero-knowledge proof of n discrete logarithms, requiring a total of $12n$ modular exponential operations. Overall, $n(5m + 12)$ modular exponential operations are required, and $m = 20$ is usually sufficient. Therefore, Protocol 2 requires $112n$ modular exponential operations, and the computation complexity is $O(n)$.

6.2. Communication Complexity

The communication complexity of a protocol is typically measured by the number of communication rounds required to complete the protocol. For example, the PSI MPC protocol described in study [15] under the malicious model requires eight rounds of communication. On the other hand, the PSI calculation under the malicious model in study [17] requires 12 rounds of communication.

In Protocol 2 proposed in this paper, the private sets of both Alice and Bob are required to be encoded into vectors, and the two parties need to conduct one round of communication to generate coding vectors using the secure ranking method in which the same numbers have the same order. In addition, Alice and Bob need to conduct four rounds of communication to implement other parts of Protocol 2. Therefore, the implementation of Protocol 2 requires five rounds of communication in total.

According to the results presented in Table 2, n represents the number of elements in the sets, it can be observed that for solving the PSI problem under the malicious model, the computational complexity of Protocol 2 is lower than that of studies [15,17], which implies that Protocol 2 is computationally more efficient. Moreover, the number of communication rounds required by Protocol 2 is lower than that of studies [15,17], indicating that the communication efficiency of Protocol 2 is higher. It can be concluded that Protocol 2 provides a more efficient and practical solution to the PSI problem under the malicious model.

Table 2. Comparison of relevant PSI protocols.

Protocol	Study [15]	Study [17]	Protocol 2
Number of participants	3	3	2
Security model	Malicious	Malicious	Malicious
Communication rounds	8	12	5
Communication complexity	$O(n\ lb\ n)$	$O(n)$	$O(n)$
Computation complexity	$O(n^2)$	$O(n\ lb^2\ n)$	$O(n)$
Key methods	Garbled Bloom filters	Oblivious Linear Function Evaluation	Paillier cryptosystem; zero-knowledge proof; cut-and-choose

It is worth noting that MPC protocols under the malicious model often require the use of cryptographic tools such as zero-knowledge proof and cut-and-choose, which may increase computation complexity and reduce efficiency. However, preprocessing or computing outsourcing can be used to improve efficiency [29].

6.3. Simulation Experiment

To assess the efficiency of Protocol 2 in this paper, a simulation was conducted using Python language on the PyCharm platform. The goal of the simulation is to compare the performance of Protocol 2 with existing protocols.

Experimental environment: Windows 10 64-bit system, Intel(R) Core(TM) i5-8400 CPU @ 2.80 GHz, 16 GB RAM.

Experimental parameter setting: The Paillier encryption scheme used in the experiment is based on 512-bit large primes p and q, resulting in a modulus $N = pq$ with a length of 1024 bits. The discrete logarithm operation uses a module p with a length of 1024 bits, and the length of the random number is set to 64 bits. The length of PSI elements used in all calculations is set to 128 bits.

When the number of set elements of each participant is 2^4, 2^5, 2^6, 2^7, 2^8, 2^9, the simulation of the private set intersection protocol under the malicious model proposed in this paper was carried out. Table 3 shows the communication traffic and runtime of it.

Table 3. Communication traffic and runtime of protocols with different numbers of set elements.

Traffic (kb)/Runtime (s) \ Set Size	2^4	2^5	2^6	2^7	2^8	2^9
Protocol 2	4.72/0.061	9.51/0.118	19.12/0.210	38.19/0.401	76.45/0.799	152.85/1.589
Study [15]	5.03/0.077	12.12/0.123	36.06/0.219	99.95/0.430	340.83/0.910	1001.51/2.110
Study [17]	3.98/0.073	7.92/0.112	15.83/0.205	31.65/0.411	63.19/0.841	126.37/1.799

As shown in Table 3, it can be concluded that Protocol 2 proposed in this paper has higher computing efficiency and lower communication traffic under the sets of different numbers of elements.

Next, the communication traffic and runtime of Protocol 2 in this paper are compared with those of related schemes under the sets with different numbers of elements. Figure 1 shows the communication traffic comparison between Protocol 2 and other existing schemes.

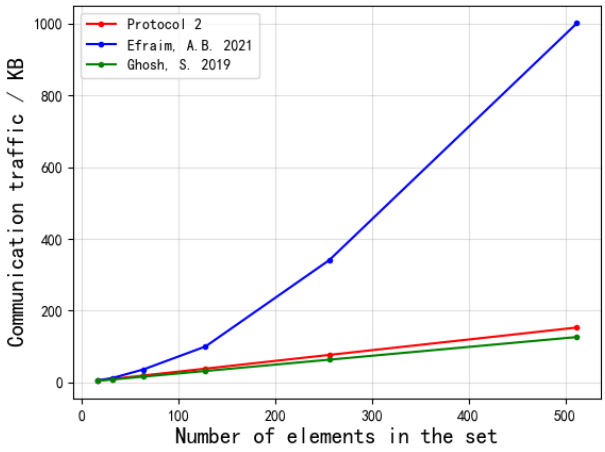

Figure 1. Communication traffic comparison with related schemes (Reference [15]: Efraim, A.B. 2021, Reference [17]: Ghosh, S. 2019).

Based on the results shown in Figure 1, it can be concluded that Protocol 2 requires less communication traffic compared to study [15], and the communication efficiency is higher, especially when the number of elements in each set is the same.

When the number of set elements is different, the runtime comparison between Protocol 2 and other existing schemes is shown in Figure 2.

Figure 2 demonstrates that the runtime of Protocol 2 increases as the number of elements in the set increases. However, when the number of elements in the set is the same, Protocol 2 has a lower runtime compared to studies [15,17], indicating higher efficiency.

Additional performance assessment was carried out through communication experiments. For this purpose, Python programs were developed using the PyCharm platform to simulate experiments with a bandwidth of 100 Mbps. The objective was to ascertain the potential delay duration during the execution of Protocol 2. It should be emphasized that, in practical settings, the delay time between different networks can vary, which might affect the performance of protocol execution. Nevertheless, this aspect was not considered in the performance evaluation. The outcomes of the communication experiments for Protocol 2 are presented in Figure 3.

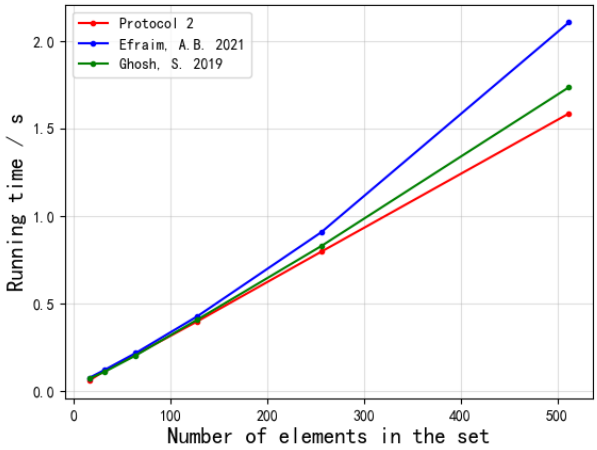

Figure 2. Runtime comparison with relevant schemes (Reference [15]: Efraim, A.B. 2021, Reference [17]: Ghosh, S. 2019).

Figure 3. The relationship between the delay time of Protocol 2 and the number of elements in the set.

The experiment demonstrates that the proposed Protocol 2 shows consistently low delay durations, which exhibit a linear increase in correspondence with the number of elements in the set. This gradual growth rate of the delay time signifies the protocol's remarkable communication efficiency.

To sum up, Protocol 2 is compared with the existing schemes through experiments, which shows that the set intersection protocol based under the malicious model in this paper is more efficient in terms of communication traffic and runtime. It is important to emphasize that while the protocol presented in this study demonstrates superior efficiency compared to existing privacy-preserving set intersection protocols, it is essential to consider practical

limitations such as restricted computational resources and communication bandwidth, as these factors could potentially impact its overall performance.

7. Conclusions

Private set intersection is considered as a significant aspect of MPC, which has diverse application scenarios. In this study, the Paillier encryption system, cut-and-choose, and zero-knowledge proof have been utilized to devise a privacy set intersection protocol under the semi-honest model. Furthermore, an MPC protocol under the malicious model has also been developed, which ensures fairness among participants and can effectively combat malicious attacks. The security of the protocol has been verified using the real/ideal model paradigm. Additionally, this protocol has demonstrated superior efficiency and practicality when compared to existing schemes.

Author Contributions: Conceptualization, X.L. and W.C.; methodology, W.C.; investigation, X.L.; software, D.L. and G.X.; experimental simulation, D.L.; security proof, D.L.; modification of English grammar, D.L.; funding acquisition, D.L.; validation, N.X. and X.C.; writing—original draft, X.L.; writing—review and editing, X.L., N.X. and X.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China: Big Data Analysis based on Software Defined Networking Architecture, grant numbers 62177019 and F0701; NSFC, grant numbers 62271070, 72293583, and 61962009; Inner Mongolia Natural Science Foundation, grant number 2021MS06006; 2023 Inner Mongolia Young Science and Technology Talents Support Project, grant number NJYT23106; 2022 Fund Project of Central Government Guiding Local Science and Technology Development, grant number 2022ZY0024; 2022 Basic Scientific Research Project of Direct Universities of Inner Mongolia, grant number 20220101; 2022 "Western Light" Talent Training Program "Western Young Scholars" Project, grant number 22040601; the 14th Five-Year Plan of Education and Science of Inner Mongolia, grant number NGJGH2021167; 2023 Open Project of the State Key Laboratory of Network and Exchange Technology, grant number 230201; 2022 Inner Mongolia Postgraduate Education and Teaching Reform Project: JGSZ2022037; the 2022 Ministry of Education Central and Western China Young Backbone Teachers and Domestic Visiting Scholars Program, grant number 2022015; Inner Mongolia Discipline Inspection and Supervision Big Data Laboratory Open Project Fund, grant number IMDBD202020; Baotou Kundulun District Science and Technology Plan Project, grant number YF2020013; Inner Mongolia Science and Technology Major Project, grant number 2019ZD025; Project JCKY2021208B036, and the Fundamental Research Funds for Beijing Municipal Commission of Education, grant number 220201.

Data Availability Statement: The authors approve that data used to support the findings of this study are included in the article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Weihong, X.I.E.; Qian, Z. The online website privacy disclosure behavior of users based on concerns-outcomes model. *Soft Comput.* **2022**, *26*, 11733–11747. [CrossRef]
2. Knott, B.; Venkataraman, S.; Hannun, A.; Sengupta, S.; Ibrahim, M.; van der Maaten, L. Crypten: Secure multi-party computation meets machine learning. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 4961–4973.
3. Zhou, J.; Feng, Y.; Wang, Z.; Guo, D. Using secure multi-party computation to protect privacy on a permissioned blockchain. *Sensors* **2021**, *21*, 1540. [CrossRef] [PubMed]
4. Yao, A.C. Protocols for secure computation. In Proceedings of the 23rd Annual Symposium on Foundation of Computer Science, Chicago, IL, USA, 3–5 November 1982; pp. 160–164.
5. Goldreich, O. *The Fundamental of Crytography: Basic Application*; Cambridge University Press: London, UK, 2004.
6. Cramer, R.; Damgard, I.B.; Nielsen, J.B. *Secure Multiparty Compution*; Cambridge University Press: London, UK, 2015.
7. Liu, J.; Tian, Y.; Zhou, Y.; Xiao, Y.; Ansari, N. Privacy preserving distributed data mining based on secure multi-party computation. *Comput. Commun.* **2020**, *153*, 208–216. [CrossRef]
8. Yao, Y.; Xiong, N.; Park, J.H.; Ma, L.; Liu, J. Privacy-preserving max/min query in two-tiered wireless sensor networks. *Comput. Math. Appl.* **2013**, *65*, 1318–1325. [CrossRef]
9. Nevo, O.; Trieu, N.; Yanai, A. Simple, fast malicious multiparty private set intersection. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, 15–19 November 2021; pp. 1151–1165.

10. Fu, A.; Zhang, X.; Xiong, N.; Gao, Y.; Wang, H.; Zhang, J. VFL: A verifiable federated learning with privacy-preserving for big data in industrial IoT. *IEEE Trans. Ind. Inform.* **2020**, *18*, 2513–2520. [CrossRef]
11. Kumar, P.; Kumar, R.; Srivastava, G.; Gupta, G.P.; Tripathi, R.; Gadekallu, T.R.; Xiong, N.N. PPSF: A privacy-preserving and secure framework using blockchain-based machine-learning for IoT-driven smart cities. *IEEE Trans. Netw. Sci. Eng.* **2021**, *8*, 2326–2341. [CrossRef]
12. Subramaniyaswamy, V.; Jagadeeswari, V.; Indragandhi, V.; Jhaveri, R.H.; Vijayakumar, V.; Kotecha, K.; Ravi, L. Somewhat homomorphic encryption: Ring learning with error algorithm for faster encryption of iot sensor signal-based edge devices. *Secur. Commun. Netw.* **2022**, *2022*, 2793998. [CrossRef]
13. Sengan, S.; Subramaniyaswamy, V.; Indragandhi, V.; Velayutham, P.; Ravi, L. Detection of false data cyber-attacks for the assessment of security in smart grid using deep learning. *Comput. Electr. Eng.* **2021**, *93*, 107211. [CrossRef]
14. Rosulek, M.; Trieu, N. Compact and malicious private set intersection for small sets. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, 15–19 November 2021; pp. 1166–1181.
15. Efraim, A.B.; Nissenbaum, O.; Omri, E.; Paskin-Cherniavsky, A. Psimple: Practical multiparty maliciously-secure private set intersection. *Cryptol. Eprint Arch.* **2021**, 122. Available online: https://eprint.iacr.org/2021/122 (accessed on 22 May 2023).
16. Liu, X.; Zhang, R.L.; Xu, G.; Chen, X.B. Securely determine the inclusion relation of a point and a convex polygon in malicious model. *J. Cryptologic Res.* **2022**, *9*, 524–534. [CrossRef]
17. Ghosh, S.; Nilges, T. An algebraic approach to maliciously secure private set intersection. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, 19–23 May 2019; pp. 154–185.
18. Rabin, M.O. How to exchange secrets with oblivious transfer. *Cryptol. Eprint Arch.* **2005**, *2005*, 187.
19. Chase, M.; Miao, P. Private set intersection in the internet setting from lightweight oblivious PRF. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 17–21 August 2020; pp. 34–63.
20. Chauhan, A.K.; Kumar, A.; Sanadhya, S.K. Quantum free-start collision attacks on double block length hashing with round-reduced AES-256. *IACR Trans. Symmetric Cryptol.* **2021**, *2021*, 316–336. [CrossRef]
21. Pinkas, B.; Rosulek, M.; Trieu, N.; Yanai, A. PSI from PaXoS: Fast, malicious private set intersection. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, 10–14 May 2020; pp. 739–767.
22. Zhang, E.; Liu, F.H.; Lai, Q.; Jin, G.; Li, Y. Efficient multi-party private set intersection against malicious adversaries. In Proceedings of the 2019 ACM SIGSAC conference on cloud computing security workshop, London, UK, 11 November 2019; pp. 93–104.
23. Yousefipoor, V.; Eghlidos, T. An efficient, secure and verifiable conjunctive keyword search scheme based on rank metric codes over encrypted outsourced cloud data. *Comput. Electr. Eng.* **2023**, *105*, 108523. [CrossRef]
24. Paillier, P. Public-key cryptosystems based on composite degree residuosity classes. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Prague, Czech Republic, 2–6 May 1999; pp. 223–238.
25. Goldreich, O.; Oren, Y. Definitions and properties of zero-knowledge proof systems. *J. Cryptol.* **1994**, *7*, 1–32.
26. Chaum, D.; Pedersen, T.P. Transferred cash grows in size. In Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques, Gold Coast, Australia, 13–16 December 1992; pp. 390–407.
27. Lindell, Y. Fast cut-and-choose-based protocols for malicious and covert adversaries. *J. Cryptol.* **2016**, *29*, 456–490. [CrossRef]
28. Goldreich, O. *Foundations of Cryptography: Volume 2, Basic Applications*; Cambridge University Press: Cambridge, UK, 2009.
29. Li, S.D.; Wang, W.L.; Du, R.M. Protocol for millionaires' problem in malicious models. *Sci. Sin. Inf.* **2021**, *51*, 75–88. [CrossRef]
30. Li, S.D.; Du, R.M.; Yang, Y.J.; Wei, Q. Secure Multiparty Multi-Data Ranking. *Chin. J. Comput.* **2020**, *43*, 1448–1462.