information

Article

A Blockchain-Based Secure Multi-Party Computation Scheme with Multi-Key Fully Homomorphic Proxy Re-Encryption

Yongbo Jiang, Yuan Zhou * and Tao Feng *

School of Computer and Communication, Lanzhou University of Technology, Lanzhou 730050, China
* Correspondence: zhouyxy@163.com (Y.Z.); fengt@lut.edu.cn (T.F.)

Abstract: At present, secure multi-party computing is an effective solution for organizations and institutions that want to derive greater value and benefit from the collaborative computing of their data. Most current secure multi-party computing solutions use encryption schemes that are not resistant to quantum attacks, which is a security risk in today's quickly growing quantum computing, and, when obtaining results, the result querier needs to collect the private keys of multiple data owners to jointly decrypt them, or there needs to be an interaction between the data owner and the querier during the decryption process. Based on the NTRU cryptosystem, which is resistant to quantum computing attacks and has a simple and easy-to-implement structure, and combined with multi-key fully homomorphic encryption (MKFHE) and proxy re-encryption, this paper proposes a secure multi-party computing scheme based on NTRU-type multi-key fully homomorphic proxy re-encryption in the blockchain environment, using the blockchain as trusted storage and a trusted execution environment to provide data security for multi-party computing. The scheme meets the requirements of being verifiable, conspiracy-proof, individually decryptable by the querier, and resistant to quantum attacks.

Keywords: secure multi-party computation; blockchain; multi-key homomorphic encryption; NTRU

Citation: Jiang, Y.; Zhou, Y.; Feng, T. A Blockchain-Based Secure Multi-Party Computation Scheme with Multi-Key Fully Homomorphic Proxy Re-Encryption. *Information* **2022**, *13*, 481. https://doi.org/10.3390/info13100481

Academic Editor: Muhammad Azeem Akbar

Received: 17 August 2022
Accepted: 4 October 2022
Published: 6 October 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

1. Introduction

With the rapid development of communication technology and the steady advancement of global information, data information owned by organizations or individuals can generate great value and wealth through communication and integration. To make full use of the value of data to obtain greater benefits, data interaction, and sharing, information integration and utilization between different institutions and organizations have become urgent needs, among which collecting data from all parties for collaborative computing is a typical scenario. Participants want to protect the security of data, protect the privacy of all parties, and ensure the fairness of computing in the process of data collection and use. The proposal and development of secure multi-party computation (SMPC) [1] provide an effective solution and technical support for the above requirements. In order to protect the privacy of all parties and the security of their private data during the use of secure multi-party computation, it is necessary to continuously improve the security of secure multi-party computation solutions. In addition, since the homomorphic encryption algorithm (FHE) [2] can solve the problem of user data privacy protection in cloud computation and big data environments, it is also a research hotspot to combine the homomorphic encryption algorithm with secure multi-party computation.

SMPC refers to the collaborative computation of a function by two or more participants in a collaborative computation, without trusting each other, using data held secretly by each party as the input. The privacy of each participant is protected by requiring each party to have no access to additional information beyond its own secret input and the

final result of the computation. Specifically, SMPC protects the privacy of each participant by means of the secret data held by the participants for $P_i, i = 1, \ldots, n$ separate secret inputs held by $x_i, i = 1, \ldots, n$, in order to jointly compute a common function f with the values of $f(x_1, x_2, \ldots, x_n) = (y_1, y_2, \ldots, y_n)$. Throughout the computation process, the participant's P_i cannot learn anything other than the result of the computation and their own secret inputs (x_i, y_i). The participants are not privy to any information other than the result of the computation and their own secret input.

Fully homomorphic encryption is an encryption algorithm that allows direct manipulation of encrypted data, which has the property that the result of the direct manipulation of the ciphertext is the same as the result of manipulating the plaintext first and then encrypting it, a property that allows it to be applied in outsourced computing scenarios. Multi-key fully homomorphic encryption (MKFHE) was first proposed by A. López-Alt et al. [3], who used a modified NTRU scheme to construct an MFHE scheme. Multi-key fully homomorphic encryption is an encryption method that can process data encrypted and uploaded by multiple different keys, breaking the restriction that homomorphic encryption can only process data encrypted by the same key, but, in the decryption method, the resultant querier needs to collect the private keys of multiple data owners to jointly decrypt the data or interaction between the data owner and the querier is required during the decryption process. To address the above issues, S. Yasuda et al. [4] proposed multi-key homomorphic proxy re-encryption (MKH-PRE). The MKH-PRE scheme allows the data owner to encrypt the data with its own public key for multi-key homomorphic computation, as well as allowing the ciphertext obtained from the homomorphic computation to be proxy re-encrypted, converting the resultant ciphertext into a new ciphertext that can only be decrypted by the resultant querier. The advantage of the NTRU cryptosystem is that it is resistant to quantum attacks, and the NTRU-based scheme is a much easier way to generate secret keys, using only modulo multiplication and modulo inverse operations, with a simple structure that is easy to implement. It can be used not only to construct NTRU-based MKFHEs but also to construct proxy re-encryption schemes that transform ciphertext into data that can be decrypted with a querier key.

1.1. Motivation and Contribution

The encryption schemes used in most current secure multi-party computing schemes are not resistant to quantum attacks, which is currently a security risk with the rapid development of quantum computing, and the SMPC scheme that uses MKFHE cannot be decrypted separately by the querier when obtaining the computation results. Therefore, it is necessary to design a secure multi-party computing scheme that is resistant to quantum attacks and can be decrypted individually by the querier.

The main research contributions of this paper are as follows:

1. A secure multi-party computation scheme based on NTRU-type [5] multi-key fully homomorphic encryption proxy re-encryption is proposed. The use of proxy re-encryption solves the problem that the multi-key homomorphic encryption scheme cannot be decrypted separately when the result is obtained, and the data owner can go offline after encrypting the uploaded data and does not have to stay online during secure multi-party computation.
2. A scheme combining the blockchain with an NTRU multi-key fully homomorphic encryption agent re-encryption secure multi-party computing scheme is proposed. The decentralized, transparent, and non-tamper characteristics of the blockchain are utilized to achieve the traceability and verifiability of the scheme and prevent collusion of the participants.
3. The security proof and comparison with other solutions demonstrate that this secure multi-party computing solution meets the requirements of being independent of trusted third parties, verifiable, privacy-protected, collusion-proof, individually decryptable by the querier, and resistant to quantum attacks.

1.2. Paper Structure

The rest of this paper is organized as follows: Section 2 presents related works. Section 3 describes the scheme model, the steps of the scheme operation, the algorithms involved, and the security model used. Section 4 proves the security of the scheme. Section 5 compares the scheme with other relevant SMPC schemes. The conclusion is provided in Section 6.

2. Related Work

YAO first proposed a two-party secure computation method in [6] using the "millionaire problem". Goldreich and others extended the two-party computation model to a basic multi-party computation model [7]. Using this as a starting point, the security of SMPC schemes has been a concern. To provide a trusted execution environment for SMPC, some researchers have chosen to perform secure multi-party computing through trusted third parties, such as Wu Y et al. who constructed a generic server-assisted secure multi-party computing protocol for secure execution of collaborative computing tasks in cloud computing [8]. However, trusted third parties are vulnerable to attacks forming a single point of failure and also have the potential to be complicit with malicious parties.

Researchers found that blockchains can provide a more secure execution environment for SMPC. The open, transparent, and tamper-evident nature of blockchain can provide a means of verification and traceability for SMPC, and the incentive mechanism can effectively prevent complicity from occurring. H. Gao et al. proposed a BFR-MPC scheme in combination with the blockchain [9] that encourages all participants to cooperate through an incentive mechanism and maintains a public reputation system in the scheme, in which honest participants gain more and more benefits while corrupt participants are increasingly punished. Y. Yang et al. proposed Block-SMPC, a blockchain-based SMPC scheme [10], which ensures data integrity and authentication by using the blockchain, introduces a multi-party computer system based on homomorphic encryption, and improves privacy security by separating the authority of homomorphic keys and ciphertexts. Liu et al. proposed a secure multi-party computing protocol, BPLSM, for ubiquitous data privacy protection in combination with blockchain technology [11]. It achieves on-chain signature verification, a guarantee of commitment, the correctness of encrypted values and address hiding, and off-chain combined transaction commitment using the property of Pederson's additive homomorphism to construct a secure multi-party computation scheme that can sign different messages in combination with the Schnorr protocol.

The secure multi-party computation scheme in the above study improves the security of SMPC with the help of blockchain features, but the scheme cannot be decrypted separately by the querier when obtaining the computation result. In order to meet the requirement of being able to carry out decryption individually, T. Wang et al. [12] proposed a secure, high-performance sharing and multi-party computing model by combining the features of the blockchain, based on a combination of on-chain storage and off-chain storage, and, in this storage environment, data are shared by using proxy re-encryption. However, most of the encryption algorithms involved in the above scheme are based on large integer decomposition or discrete logarithm difficulty problems, which do not have the ability to resist quantum attacks.

To solve these problems, this paper proposes a secure multi-party computing scheme based on a multi-key homomorphic proxy re-encryption scheme and an NTRU-based MKFHE scheme [13,14] with resistance to quantum attacks in the blockchain environment.

3. SMPC Scheme with Multi-Key Fully Homomorphic Proxy Re-Encryption

3.1. System Model

The system consists of several components: the data owner, the data querier (in general, the data owner, but possibly also the authorized user), the computation network, the SMPC contract, the InterPlanetary File System (IPFS) [15], and the blockchain. The system architecture is shown in Figure 1. The functions of each part of the system are as follows:

- Data owner

As the data provider of secure multi-party computation, the data owner owns the original data as the input of the computation. To ensure the privacy and data security of all parties, the data must be encrypted by the data owner before being used as the input of the computation.

- Result inquirer

As the receiver of the computation result, the result inquirer is generally the data owner or the authorized user who does not provide the data. With the support of the proxy re-encryption algorithm, the result inquirer can decrypt the encrypted computation result through their own private key and obtain the calculation result.

- Blockchain

The blockchain participates in the process as a trusted storage and execution environment. This scheme provides resistance to quantum attacks through proxy re-encryption to enable the result querier to decrypt the ciphertext result alone. At the same time, open, transparent, and untampered information stored on the blockchain can be verified as proof.

- IPFS

IPFS is used to store encrypted raw data as off-chain storage to save storage space. A Bloom filter [16] generates index values, and then IPFS uploads the data keywords, index values, and storage address to the blockchain. SMPC nodes look up the data storage address on the blockchain and then download the encrypted data from IPFS to local storage for calculation.

- SMPC Contracts

Data owners, data inquirers, and SMPC nodes need to register with the SMPC contract before the calculation begins. Participants (SMPC nodes or users) pay a deposit to the SMPC contract, and the SMPC contract returns a unique ID to the registrant. The data inquirers send their public keys to the SMPC contract, which generates the proxy re-encryption key. The computation function in the contract is agreed upon in advance by the participants of the secure multi-party calculation so that the code can be written and deployed on the blockchain platform to automatically trigger the execution of the agreed computation without human intervention.

- Computing networks

The SMPC computing network undertakes the task of data calculation. It queries the corresponding encrypted data on IPFS as the input and performs the calculation on the encrypted data consistently with the agreed calculation function according to the SMPC contract. The obtained encryption results are sent to each data interrogator after the agent re-encryption operation.

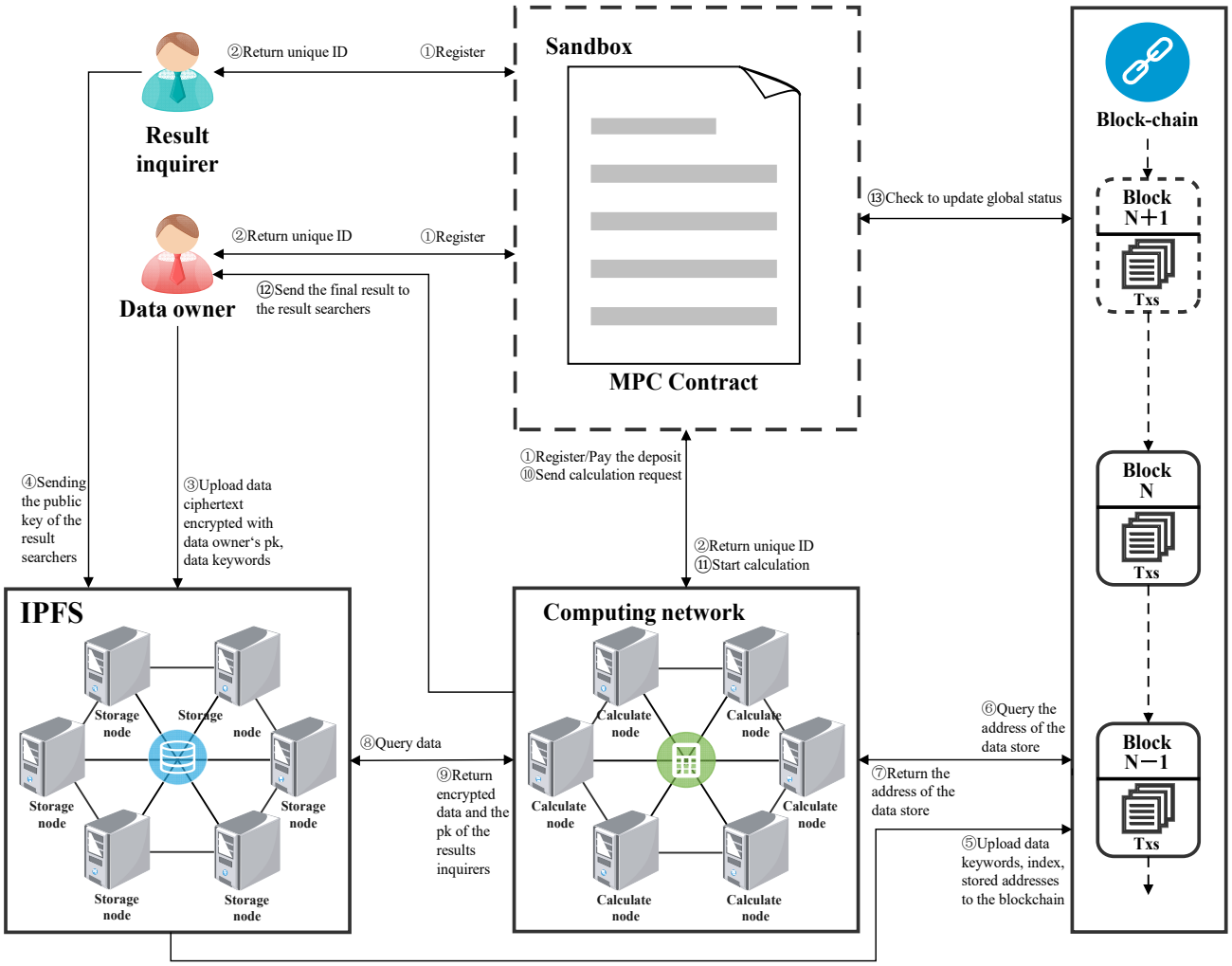

Figure 1. System model diagram.

3.2. Program Steps

The steps in the operation of the system are shown in Figure 2.

1. Initially, the data owner, the data querier, and the SMPC node register with the SMPC contract, which distributes a unique ID to each registered node, while the SMPC node pays a deposit to the contract.
2. The data owner generates keywords for the original data to be involved in the operation and then encrypts the data to be involved in the operation with its own public key and uploads them to IPFS, where a Bloom filter generates the index value of the encrypted data. The data owner uploads the keywords generated from the original data and the storage address of the encrypted data.
3. The computing network node interacts with the blockchain by querying keywords, querying the corresponding block to obtain the storage address of the required encrypted information, and obtaining the encrypted data from the IPFS data storage address for calculation.
4. The data querier sends its public key to the SMPC contract, and the ciphertext result after the homomorphic calculation is converted into the ciphertext result encrypted by the data querier's public key through the NTRU proxy re-encryption algorithm. To obtain the final calculation result, the data querier only needs to decrypt the calculation result returned by the computing network with its own private key. The contract is carried out in a sandbox isolation environment, and the blockchain rewards or deducts the deposit based on whether the node is honest or not.

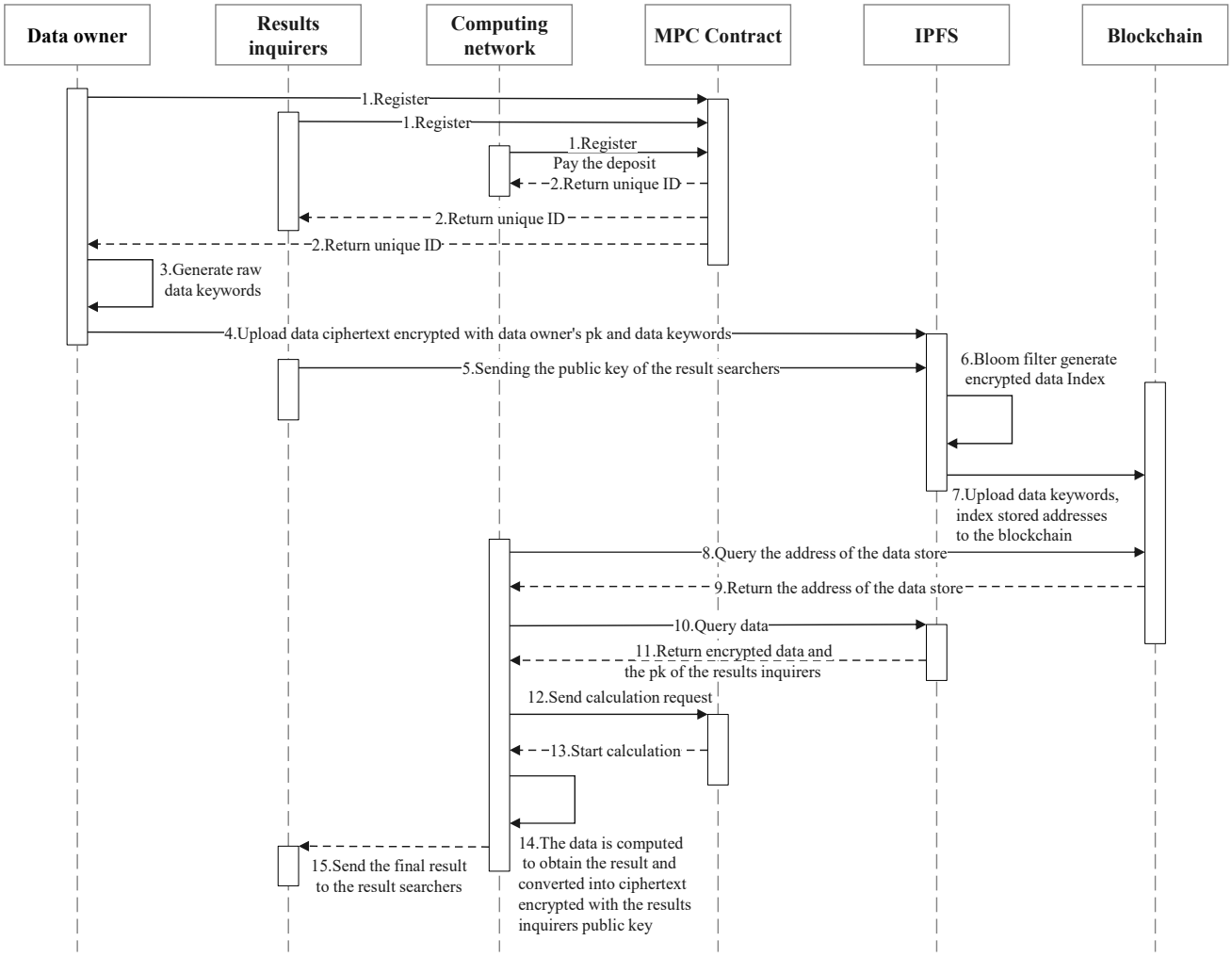

Figure 2. Timing diagram of system operation.

5. At the end of the calculation, the blockchain validation node checks whether any dishonest nodes have committed mischief before or after the calculation process. If this does not happen, the deposit of each node will be returned as is; if this happens, the deposit of the honest node will be returned, and the deposit of the dishonest node will be deducted and distributed as a reward to the honest node as a punishment.

In order to enable the nodes in the scheme to reach consensus quickly while ensuring security during the operation, the Score Grouping-practical Byzantine fault-tolerant (SG-PBFT) consensus algorithm, which is based on a modified version of the practical Byzantine fault-tolerant (PBFT) algorithm [17] proposed in the literature [18], is used in the scheme.

The SG-PBFT sets the initial score of N sequential random nodes as 100 points and divides them into a consensus node set and a candidate node set. The consensus node executes the consensus process, while the candidate node does not participate in the consensus process and only receives the consensus results. The primary node is selected by $p = v \bmod CN$. CN represents the agreed number of nodes. When the primary node p is attacked or fails, the view v will be changed, and the recalculated primary node will replace it.

When the nodes reach a consensus, the master node will send the confirmed results to all consensus nodes and update the score of the node. If the result of the node is consistent with the consensus result, one point will be added. Otherwise, five points will be deducted. The m nodes with the lowest score will be removed from the consensus node

set and attached to the end of the candidate node set. The m nodes with the highest score in the candidate set will be added to the consensus node set and renumbered.

The SG-PBFT renumbers and adjusts nodes after each agreement is reached. This ensures that the identity of the primary node is hidden and therefore resistant to distributed denial of service (DDoS) attacks. In the SG-PBFT, even if all malicious nodes join together, they can only send no more than 1/3 of the total number of messages. Malicious nodes cannot reach a consensus, therefore the SG-PBFT can resist selective attacks. The SG-PBFT operating process is shown in Figure 3. The "x" on the line indicates that the node is a failed node.

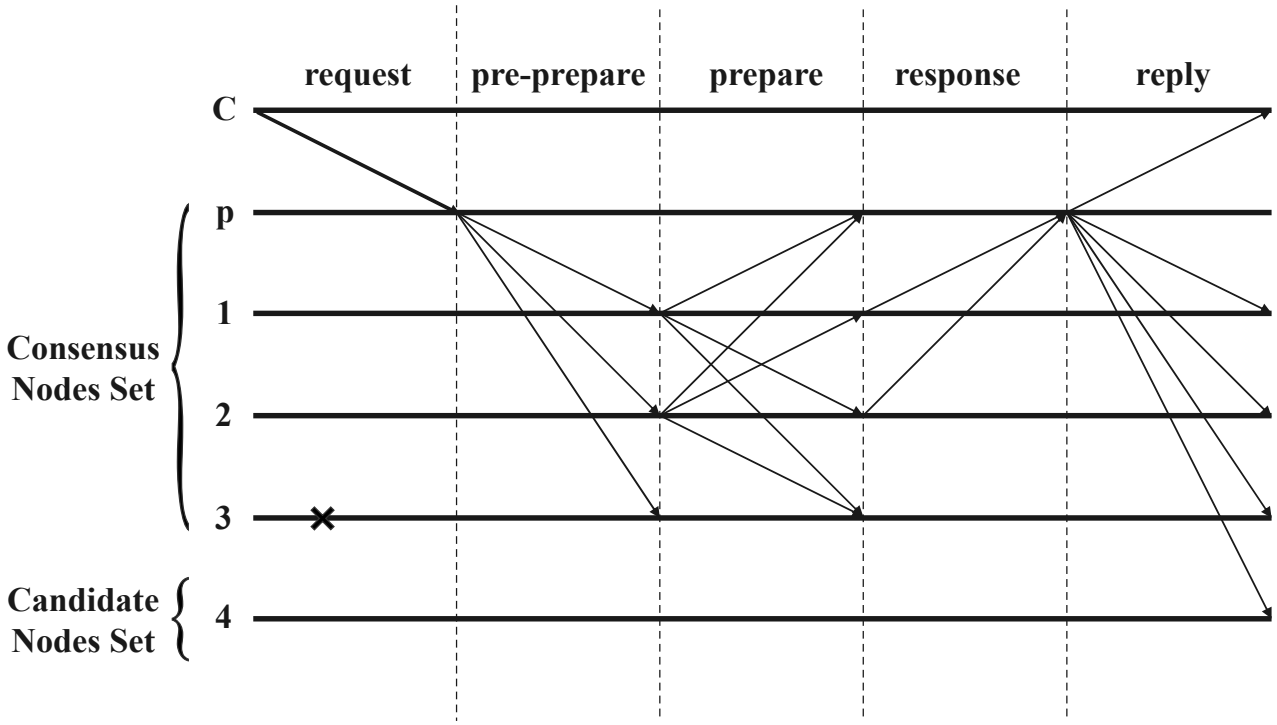

Figure 3. SG-PBFT operating diagram.

3.3. Algorithm Construction

The algorithms in the scheme are divided into four parts: the initialization algorithm, the key generation algorithm, the multi-key homomorphic encryption algorithm, and the proxy re-encryption algorithm. The operations in the scheme are all performed on the ring $R = \mathbb{Z}[X]/\Phi_m(x)$, over which q is a prime number, and $\Phi_m(x)$ is a partitioned circle polynomial of the degree $n = \varphi(m)$. Let $R_q = R/qR$.

- Initialization algorithm

Setup$(1^\lambda) \longrightarrow pp$: Enter the security parameters' λ to generate ring learning with errors (RLWE) [19] with dimension n, the plaintext modulus p, the ciphertext modulus q, and the ring R distribution over the ring ψ, φ. Vector $\boldsymbol{\alpha}$ and matrix $\boldsymbol{B}$ are extracted randomly on the uniform distributions $U(R_q^d)$ and $U(R_q^{2\times m})$ of ring R, respectively, and then a common parameter $pp = (n, p, q, \psi, \varphi, \boldsymbol{\alpha}, \boldsymbol{B})$ is output.

The public parameter pp is used as the input to the key generation algorithm for the generation of keys by the data owner and the result querier in addition to the generation of the computational key **Evk** in the multi-key homomorphic encryption algorithm.

- Key generation algorithm

KeyGen$(pp) \longrightarrow (pk_i, sk_i)$: Randomly select u', l in the distribution ψ; it is required that the extracted u' is reversible in R_q. Calculate $u = pu' + 1(\mathrm{mod}q)$, $u^{-1} \in R_q$, and let $v = plu^{-1}(\mathrm{mod}q)$. The error vector $\boldsymbol{\beta}$ is randomly selected from the distribution φ^d, and

$\boldsymbol{f} = -\boldsymbol{\alpha}u + \boldsymbol{\beta}(\mathrm{mod}q)$ is calculated. Set $u = (u, 1) \in R_q^{1\times 2}$, randomly select the error vector $\boldsymbol{\beta}'$ from the distribution $\varphi^{d'}$, and calculate g= $-\boldsymbol{uB} + \boldsymbol{\beta}'(\mathrm{mod}q) \in R_q^{n'}$. Output the private key $sk = u$ and the public key $pk = (v, \boldsymbol{f}, \boldsymbol{g})$.

The data owner and the result querier generate their respective public key pk and private key sk by means of a key generation algorithm.

- Multi-key homomorphic encryption algorithm

The components of the multi-key homomorphic encryption algorithm include the computational key generation algorithm EvkGen, the encryption algorithm Enc, the ciphertext extension algorithm CtxtExtend, and the homomorphic computation algorithm Eval.

EvkGen$(pp, pk, sk) \rightarrow$ **Evk**: Randomly select s in the distribution φ, randomly select $\boldsymbol{r}_0, \boldsymbol{\beta}_0, \boldsymbol{\beta}_1$ in the distribution φ^d, calculate $\mathbf{evk}_0 = vr_0 + \boldsymbol{\beta}_0 + u^{-1}sl(\mathrm{mod}q)$ and $\mathbf{evk}_1 = s\boldsymbol{\alpha} + \boldsymbol{\beta}_1 + ul(\mathrm{mod}q)$, and output the computation key $\mathbf{Evk} = [\mathbf{evk}_0|\mathbf{evk}_1]$.

The computational key generation algorithm EvkGen generates the computational key **Evk** using the public parameters, the public key pk of the data owner, and the private key sk.

Enc$(pk, m) \longrightarrow c$: Randomly selects r, β in the distribution φ, set $\delta = \lfloor q/p \rfloor$, and compute the ciphertext $c = vp + \beta + \delta m(\mathrm{mod}p)$.

The data owner uses the encryption algorithm Enc to encrypt the data they need to participate in the operation and then generates a cipher text and uploads it to IPFS.

CtxtExtend$(\boldsymbol{c}_1, \boldsymbol{c}_2, .., \boldsymbol{c}_s) \rightarrow \boldsymbol{c}_i^*$: Let the number of ciphertexts involved in this operation be s and $\boldsymbol{c}_1 = (c_1, c_2, .., c_{k_i}) \in R_q^{k_i}$. The corresponding user ID set is $\{id_{k_1}, id_{k_2}, .., id_{k_i}\}, i = 1, 2, \ldots, s$. Let $k = \max(k_1, k_2, \ldots, k_s)$ and output the k-dimensional ciphertext $\boldsymbol{c}_i^* = (c_1^*, c_2^*, \ldots, c_k^*) \in R_q^k$ where

$$c_i^* = \begin{cases} c_j, i = id, 1 \le j \le k_i \\ 0, \text{others} \end{cases} \quad (1)$$

Eval$(\boldsymbol{c}_1, \boldsymbol{c}_2, \mathbf{Evk})$: Calling CtxtExtend$(\boldsymbol{c}_1, \boldsymbol{c}_2)$ gives $\boldsymbol{c}_1^*, \boldsymbol{c}_2^* \in R_q^k$ by calculation. Then homomorphic addition or multiplication is performed.

1. HAdd$(\boldsymbol{c}_1^*, \boldsymbol{c}_2^*)$: Compute and output the ciphertext $\boldsymbol{c} = \boldsymbol{c}_1^* + \boldsymbol{c}_2^* (\mathrm{mod}q)$.
2. HMult$(\boldsymbol{c}_1^*, \boldsymbol{c}_2^*, \mathrm{evk})$: Calculate $\boldsymbol{c}' = \lfloor p/q \times \boldsymbol{c}_1^* \otimes \boldsymbol{c}_2^* \rceil (\mathrm{mod}q)$ and output the ciphertext $\boldsymbol{c} = \mathrm{Reline}(\boldsymbol{c}', \mathrm{evk}) \in R_q^k$.

- Proxy re-encryption algorithm

Proxy re-encryption algorithms include the re-encryption key generation algorithm RKGen, the re-encryption algorithm ReEnc, and the re-encrypted ciphertext decryption algorithm PRDec. The algorithms are used to re-encrypt the ciphertext after the homomorphic encryption calculation into a ciphertext that can be decrypted by the resultant querier's private key.

RKGen$(sk_i, pk_j) \rightarrow rk_{i\rightarrow j}$: Select pk_j from $\boldsymbol{h}_j$ and let $\boldsymbol{H}_j = \boldsymbol{B} + \boldsymbol{\beta}_2^T \otimes \boldsymbol{h}_j$, where $\boldsymbol{\beta}_2 = (0 \parallel 1) \in \mathbb{Z}^2$. Select X_i, $n' \ge 2\log q + 2\lambda$, from $\{0,1\}^{n'\times d}$ and compute and output $rk_{i\rightarrow j} = \mathbf{RK}_{i\rightarrow j} = \boldsymbol{H}_j, \boldsymbol{X}_i + \begin{pmatrix} \mathbf{0} \\ u_i \boldsymbol{l} \end{pmatrix} (\mathrm{mod}q) \in R_q^{2\times h}$.

ReEnc$(rk_{1\rightarrow j}, \ldots, rk_{k\rightarrow j}, c) \rightarrow c'$: Let $\mathbf{c} = (c_1, .., c_k)$ and calculate and output $c' = \sum_{i=1}^k RK_{i\rightarrow j} u^{-1}(c_i) (\mathrm{mod}q) \in R_q^2$.

PRDec$(sk_j, c') \rightarrow$ m : Let $\boldsymbol{u}_j = (u_j, 1)$ and compute and output $\mathrm{m} = \lfloor (p/q)\langle \boldsymbol{u}_j, c' \rangle \rceil (\mathrm{mod}p)$.

3.4. Security Model

The scheme uses the definition of security from the literature [5] for the MKH-PRE scheme. The definition designs an IND-CPA security game between a challenger and an adversary $\mathcal{A}$. The re-encryption process is represented using a directed acyclic graph, such that D is the set of edges in the re-encryption graph. During the game, the adversary can initiate an interrogation of the challenger about the re-encryption key generation based on the re-encryption graph. The formal definition of a secure game is as follows:

- Preparation phase

The challenger sends the generated public parameters $\mathrm{Setup}(1^{\lambda}) \to pp$ to the adversary $\mathcal{A}$.

Generate honest keys. The number of honest keys received by the challenger from $\mathcal{A}$ is N_{h}, and the challenger generates $(pk_i, sk_i), i = 1, \ldots, N_{\mathrm{h}}$ and sends the pk_i to $\mathcal{A}$. Let χ_{h} be the set of honest public keys.

Generate non-honest keys. The number of non-honest keys received by the challenger from $\mathcal{A}$ is N_d, and the challenger generates $(pk_i, sk_i), i = 1, \ldots, N_d$ and sends the pk_i to $\mathcal{A}$. Let $\chi_{\boldsymbol{d}}$ be the set of non-honest public keys.

- Inquiry phase

The adversary can initiate a polynomial inquiry of any order.

Generate the re-encryption key. $\mathcal{A}$ sends (i, j) to the challenger. If $i, j \in \chi_{\mathrm{h}}$, and there is a directed acyclic graph $E = (\chi_{\mathrm{h}}, D \cup (i, j))$, then the challenger adds (i, j) to D and sends the generated re-encryption key for i to j $\mathrm{RKGen}(sk_i, pk_j) \to rk_{i \to j}$ to $\mathcal{A}$; otherwise, $\perp$ is returned.

Re-encryption. $\mathcal{A}$ sends (i, j, c) to the challenger. If $i, j \in \chi_{\boldsymbol{d}}$, and $= c'$, the challenger returns $\perp$. Otherwise, the challenger sends a ciphertext re-encrypted with the j's public key c_j sent to $\mathcal{A}$; otherwise, $\perp$ is returned.

- Challenge phase.

The plaintext space is M. Take $m_0, m_1 \in M$, and $i' \in N_{\mathrm{h}}$. $\mathcal{A}$ sends (i', m_0, m_1) to the challenger, who chooses a random bit $b \in \{0,1\}$, generates c' by $\mathrm{Enc}(pk_{i'}, m_b)$, and returns it to $\mathcal{A}$. $\mathcal{A}$ can only initiate a challenging inquiry once.

- Judgment phase.

$\mathcal{A}$ outputs one bit $b' \in \{0,1\}$. In this game, the advantage of adversary $\mathcal{A}$ is defined as

$$\mathrm{Adv}^{\mathrm{ind-cpa}}_{\mathrm{MKH-PRE},\mathcal{A}}(\lambda) = \left| \Pr[b' = b] - \frac{1}{2} \right| \tag{2}$$

If, for any probabilistic polynomial time adversary $\mathcal{A}$ there is

$$\mathrm{Adv}^{\mathrm{ind-cpa}}_{\mathrm{MKH-PRE},\mathcal{A}}(\lambda) = \mathrm{negl}(\lambda) \tag{3}$$

then, the scheme is IND-CPA safe.

4. Proof of Safety

Here, we demonstrate the safety of the MKH-PRE scheme. If the RLWE assumption, the decisional small polynomial ratio (DSPR) [20] assumption, and the cyclic safety assumption are difficult, then the MKHE scheme in this paper is IND-CPA safe.

The security of the PRE process is considered below. Here, we demonstrate the security of the PRE process through an IND-CPA security game between a challenger and an adversary $\mathcal{A}$. $\mathcal{A}$ is an adversary in arbitrary probabilistic polynomial time, which has access to the re-encryption key generation and evolution RKGen and the re-encryption oracle machine ReEnc and can only initiate queries for generating re-encryption keys based on the re-encryption graph. Consider the following set of security games:

Game 0. The IND-CPA safe game was defined in the previous section. Assuming $\chi_{\mathrm{h}} = \{1, \ldots, N\}$, $\chi_{\mathrm{b}} = \{N + 1, \ldots, M\}$. According to the topological order determined by the

re-encryption graph, if $i < j$, then there are no edges from i to j, i.e., $\mathcal{A}$ can only be initiated in the $i > j$ case of a re-encryption key $rk_{i\to j}$ of the query.

Divide **Game k**, $k = 1, \ldots, N$, into two categories, **Game. 1 k** and **Game. 2 k**.

Game. 1 k. When $\mathcal{A}$ initiates a query to generate an honest key, for all $i < k$, the challenger randomly draws v_i and $\boldsymbol{g}_i$ in the uniform distributions R_q and $R_q^{n'}$, respectively, to generate the public key; for all $k < i \le N$, the challenger generates the public key by $\mathrm{KeyGen}(pp) \longrightarrow (v_i, \boldsymbol{g}_i)$. The rest of the operation is the same as Game. 2 k-1.

Game. 2 k. When A initiates the query to generate the re-encryption key, the challenger generates the re-encryption key $rk_{i\to j}$ by drawing a random matrix from $R_q^{2\times n'}$ for all $j < i \le k$; for $k < i, j \le N$, the challenger generates the re-encryption key by $\mathrm{RKGen}(sk_i, pk_j) \to rk_{i\to j}$. The rest of the operation is the same as **Game. 2 k**.

Game End. When $\mathcal{A}$ initiates a challenge query, the challenger generates the ciphertext c through random sampling, and the rest of the operation is the same as **Game. 2 N.**

The strengths of $\mathcal{A}$ in each game are assessed separately as follows:

Because **Game 0** is an IND-CPA safe game of the original MKH-PRE scheme,

$$\mathrm{Adv}^{\mathrm{ind-cpa}}_{\mathrm{MKH-PRE},\mathcal{A}}(\lambda) = \mathrm{Adv}^{\mathrm{Game\ 0}}_{\mathcal{A}}(\lambda) \tag{4}$$

In **Game. 1 k**, the re-encryption key $rk_{k\to i}$ generated by the challenger satisfies $rk_{k\to i} = \mathbf{RK}_{k\to i} = \boldsymbol{H}_i, \boldsymbol{X}_k + \begin{pmatrix} \mathbf{0} \\ u_k \boldsymbol{l} \end{pmatrix} (\mathrm{mod} q)$ when $i < k$, where $\boldsymbol{X}_k$ is randomly selected in the uniform distribution of $\{0,1\}^{n'\times d}$ and $n' \ge 2\log q + 2\lambda$. Since $\boldsymbol{h}_i$ and $\boldsymbol{B}$ are randomly selected from the uniform distribution when $i < k$, $\boldsymbol{H}_i = \boldsymbol{B} + \boldsymbol{\beta}_2^T \otimes \boldsymbol{h}_i$ is also subject to the uniform distribution. According to the residual hash lemma, H and X are subject to uniform distributions, so $\boldsymbol{H}_i$ and $\boldsymbol{X}_k$ are statistically indistinguishable from a matrix randomly drawn from a uniform distribution. The results show that $rk_{k\to i}$ is statistically indistinguishable from the random matrix extracted from the uniform distribution, meaning that **Game. 1 k** and **Game. 2 k** are statistically indistinguishable. Therefore, there is

$$\left|\mathrm{Adv}^{\mathrm{Game.1}\ k}_{\mathcal{A}}(\lambda) - \mathrm{Adv}^{\mathrm{Game.2}\ k}_{\mathcal{A}}(\lambda)\right| = \mathrm{negl}(\lambda) \tag{5}$$

Adversary $\mathcal{A}$ constructs a PPT algorithm $\mathcal{Q}$ to distinguish the RLWE distribution from the uniform distribution, and the sample $\boldsymbol{x} \in R^2$ input to $\mathcal{Q}$ comes from the RLWE distribution or one of the uniform distributions.

- Preparation phase

$\mathcal{Q}$ calculates $\overline{\boldsymbol{B}} = (x_1^{\mathrm{T}} \| \ldots \| x_{n'}^{\mathrm{T}})$ and randomly extracts $v_k, \boldsymbol{g}_k$ from the uniform distribution $R_q, R_q^{n'}$. Q sends $\boldsymbol{B} = \overline{\boldsymbol{B}} + \begin{pmatrix} \mathbf{0} \\ \boldsymbol{d}_{\mathbf{k}} \end{pmatrix}$ to $\mathcal{A}$.

Generate honest keys

When $\mathcal{A}$ initiates an honest key generation query, the response of $\mathcal{Q}$ is as follows:

When $i < k$, $\mathcal{Q}$ randomly selects $v_i, \boldsymbol{g}_i$ from the uniform distribution $R_q, R_q^{n'}$ and lets $pk_i = (v_i, \boldsymbol{g}_i)$.

When $i = k$, $\mathcal{Q}$ lets $pk_i = (v_k, \boldsymbol{g}_k)$.

When $i > k$, $\mathcal{Q}$ generates the public key by $\mathrm{KeyGen}(pp) \longrightarrow (pk_i, sk_i)$.

Finally, $\mathcal{Q}$ sends pk_i, $i \in \{1, \ldots, k\}$ to $\mathcal{A}$.

Generate dishonest keys

When $\mathcal{A}$ initiates a generate dishonest keys query, $\mathcal{Q}$ calculates $\mathrm{KeyGen}(\boldsymbol{B}) \longrightarrow (pk_i, sk_i)$ and sends (pk_i, sk_i) to $\mathcal{A}$.

- Inquiry phase.

Generate re-encryption keys

When $\mathcal{A}$ initiates a generate re-encryption key query (i, j), if $i, j < k$, then $\mathcal{Q}$ returns $\mathbf{RK}_{i\to j} \in R_q^{2\times h}$ randomly selected from the uniform distribution; if $i, j > k$, then $\mathcal{Q}$ returns $\mathrm{RKGen}(sk_i, pk_j) \to \mathbf{RK}_{i\to j}$.

Re-encryption

When $\mathcal{A}$ initiates a re-encryption query $(i_1, \ldots, i_s, j, c)$, $\mathcal{Q}$ returns $\mathrm{ReEnc}(\mathbf{RK}_{i_1\to j}, \ldots, \mathbf{RK}_{i_s\to j}, \mathrm{c}) \to \mathrm{c}_j$ to $\mathcal{A}$.

- Challenge phase.

$m_0, m_1 \in M$, and $i' \in N_h$ are selected. $\mathcal{A}$ starts the query (i', m_0, m_1), and Q randomly selects a bit $b \in \{0,1\}$, generating c' by $\mathrm{Enc}(pk_{i'}, m_b)$ and returning it to $\mathcal{A}$.

- Judgment phase.

$\mathcal{A}$ ceases the query and outputs bits $b' \in \{0,1\}$. Then, Q outputs 1 if $b = b'$, or 0 otherwise.

If the RLWE distribution is entered in Q, then Q simulates Game. 2 k−1. The first rows of $\bar{\boldsymbol{B}}$ and $\boldsymbol{g}_k$ are both random quantities that obey a uniform distribution, so $\boldsymbol{B}$ also obeys a uniform distribution. In addition, the distributions of $\boldsymbol{g}_k \approx -(f||1)\boldsymbol{B}$ are the same as in the actual game. In the previous game, $rk_{i\to j}$ was replaced by a randomly selected matrix from a uniform distribution, where $\boldsymbol{B}$ simulates Game. 2 k-1. If the uniform distribution is entered in Q, then $\boldsymbol{B}$ simulates Game.1 k. From the above analysis, the RLWE assumption, and the DSPR assumption, it follows that:

$$\left|\mathrm{Adv}_{\mathcal{A}}^{\mathrm{Game.2}\ k-1}(\lambda) - \mathrm{Adv}_{\mathcal{A}}^{\mathrm{Game.1}\ k}(\lambda)\right| = \mathrm{negl}(\lambda) \tag{6}$$

The ciphertext in **Game End** is drawn randomly from a uniform distribution; in **Game. 2 N**, all public keys are replaced in the previous games with vectors drawn randomly from a uniform distribution, and the polynomial drawn randomly from a uniform distribution is statistically indistinguishable from the ciphertext output by the encryption algorithm under the RLWE assumption; therefore, **Game. 2 N** is statistically indistinguishable from **Game End**.

$$\left|\mathrm{Adv}_{\mathcal{A}}^{\mathrm{Game.2}\ N}(\lambda) - \mathrm{Adv}_{\mathcal{A}}^{\mathrm{Game\ End}}(\lambda)\right| = \mathrm{negl}(\lambda) \tag{7}$$

The advantages of Q in **Game End** are:

$$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{Game\ End}}(\lambda) = \mathrm{negl}(\lambda) \tag{8}$$

Based on the above analysis, it can be concluded that

$$\begin{aligned}(\mathrm{Adv}_{\mathrm{MKH-PRE},\mathcal{A}}^{\mathrm{ind-cpa}}(\lambda) &\\ \leq \sum_{k=1}^{N}\left|\mathrm{Adv}_{\mathcal{A}}^{\mathrm{Game.2}\ k-1}(\lambda) - \mathrm{Adv}_{\mathcal{A}}^{\mathrm{Game.1}\ k}(\lambda)\right| &+ \left|\mathrm{Adv}_{\mathcal{A}}^{\mathrm{Game.2}\ N}(\lambda) - \mathrm{Adv}_{\mathcal{A}}^{\mathrm{Game\ End}}(\lambda)\right| \\ + \mathrm{Adv}_{\mathcal{A}}^{\mathrm{Game\ End}}(\lambda) = \mathrm{negl}(\lambda) &\end{aligned} \tag{9}$$

Therefore, the solution in this paper is IND-CPA safe.

5. Comparison of Programs

Constructed by Y. Wu et al. to address the problem of secure execution of collaborative computing tasks in cloud computing, a generic server-assisted secure multi-party computing protocol [8] was proposed without complicity between the server and client; however, the scheme also relies on the participation of trusted third parties, cannot prevent complicity, and does not have a reliable means of verification of the computing process. To make the scheme verifiable, researchers have used the properties of the blockchain to add verifiability to the scheme while providing privacy protection by introducing the blockchain in the scheme construction. Examples include H. Gao et al.'s blockchain-based BFR-MPC scheme [9], Y. Yang et al.' s SMPC scheme Block-SMPC [10], and Liu et al.'s BPLSM [11], a secure multi-party computing protocol for ubiquitous data privacy protection combined with the blockchain technology. The above schemes make improvements to the scheme via the blockchain but require joint decryption by multiple querying parties when obtaining the results. T. Wang et al. [12] proposed a sharing and multi-party computation mode scheme in combination with the blockchain, using proxy re-encryption for data sharing so that querying parties can carry out decryption individually, but none of the encryption methods used in the above schemes are resistant to quantum attacks.

This scheme performs multi-party secure computation on the blockchain and pays a deposit during the computation of a transaction via a smart contract to prevent complicity between participants. The NTRU-based scheme design provides the scheme with resistance to quantum attacks. The decryption of ciphertext results by the result querier alone is achieved through proxy re-encryption, and the data owner can go offline after uploading the data and public key. The specific performance comparison is shown in Table 1. As can be seen from the table, the scheme can meet the requirements of not relying on trusted third parties, being verifiable, having privacy protection, being anti-complicity, being individually decryptable by the querier, and being resistant to quantum attacks.

Table 1. Performance comparison of this paper with other SMPC solutions.

Literature	Not Relying on Trusted Third Parties	Verifiable	Privacy	Anti-Conspiracy	Individual Decryption Available to Enquirers	Resistant to Quantum Attacks
[8]	×	×	√	×	×	×
[9]	√	√	√	√	×	×
[10]	√	√	√	√	×	×
[11]	√	√	√	-	×	×
[12]	√	√	√	-	√	×
This paper	√	√	√	√	√	√

In the table, "√" means that the scheme can meet this requirement, "×" means that the scheme cannot meet this requirement, and "-" means that the literature does not describe whether the scheme can meet this requirement.

6. Conclusions

In this paper, to solve the problem that the encryption schemes used in most current secure multi-party computation schemes are not resistant to quantum attacks, and the secure multi-party computation schemes constructed via MKFHE cannot be decrypted by the result querier alone when the result is obtained, an NTRU-type multi-key fully homomorphic proxy re-encryption secure multi-party computation scheme in the blockchain environment was proposed. By designing a multi-key fully homomorphic encryption algorithm and a proxy re-encryption algorithm under the NTRU cryptosystem, the scheme meets the requirements of individual decryption by the querier, offline access by the data owner after uploading encrypted data, and resistance to quantum attacks. At the same time, the decentralized, immutable, open, and transparent nature of the blockchain provides a trusted execution environment for the scheme, providing a traceable and verifiable means of data. The blockchain's incentives encourage honest cooperation between the various computing participants and prevent complicity.

The security of the scheme is based on the RLWE problem and the DSPR assumption, which is not a standard cryptographic assumption. Although there is no efficient way to break the DSPR assumption for the small-modulus case, it can be assumed that the DSPR assumption is secure, but this issue needs attention. Therefore, how to construct a secure multi-party computation scheme with NTRU-type multi-key fully homomorphic proxy re-encryption whose security depends only on the RLWE problem requires further research. In addition, how to apply the scheme proposed in this paper in the actual multi-party computation scenario is also an important research direction.

Author Contributions: Conceptualization, Y.J. and Y.Z.; formal analysis, Y.J., Y.Z., and T.F.; supervision, Y.J., Y.Z., and T.F.; validation, Y.J. and Y.Z.; writing—original draft, Y.J. and Y.Z.; writing—review and editing, T.F. 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, grant numbers 62162039 and 61762060, and the Foundation for the Key Research and Development Program of Gansu Province, China, grant number 20YF3GA016.

Data Availability Statement: Not applicable.

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

References

1. Yang, Y.; Huang, X.; Liu, X. A Comprehensive Survey on Secure Outsourced Computation and Its Applications. *IEEE Access* **2019**, *7*, 159426–159465. https://doi.org/10.1109/access.2019.2949782.
2. Akl, S.G.; Assem, I. Fully homomorphic encryption: A general framework and implementations. *Int. J. Parallel Emergent Distribted Syst.* **2020**, *35*, 493–498. https://doi.org/10.1080/17445760.2018.1553041.
3. López-Alt, A.; Tromer, E.; Vaikuntanathan, V. Multikey fully homomorphic encryption and applications. *SIAM J. Comput.* **2017**, *46*, 1827–1892. https://doi.org/10.1137/14100124x.
4. Yasuda, S.; Koseki, Y.; Hiromasa, R. Multi-key homomorphic proxy re-encryption. International Conference on Information Security, Guildford, UK, 9–12 September 2018; Springer: Berlin/Heidelberg, Germany, 2018; pp. 328–346. https://doi.org/10.1007/978-3-319-99136-8_18.
5. Stehlé, D.; Steinfeld, R. Making NTRU as secure as worst-case problems over ideal lattices. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, 15–19 May 2011; Springer: Berlin/Heidelberg, Germany, 2011; pp. 27–47. https://doi.org/10.1007/978-3-642-20465-4_4.
6. Yao, A.C. Protocols for secure computations. In Proceedings of the 23rd Annual Symposium on Foundations of Computer Science, Chicago, IL, USA, 3–5 November 1982; pp. 160–164. https://doi.org/10.1109/sfcs.1982.38.
7. Micali, S.; Goldreich, O.; Wigderson, A. How to play any mental game. In Proceedings of the Nineteenth ACM Symp. on Theory of Computing, STOC. 1987; ACM New York, NY, USA, 25-27 May 1987; pp. 218–229. https://doi.org/10.1145/28395.28420.
8. Wu, Y.; Wang, X.; Susilo, W. Generic server-aided secure multi-party computation in cloud computing. *Comput. Stand. Interfaces* **2022**, *79*, 103552. https://doi.org/10.1016/j.csi.2021.103552.
9. Gao, H.; Ma, Z.; Luo, S.; Wang, Z. BFR-MPC: A Blockchain-Based Fair and Robust Multi-Party Computation Scheme. *IEEE Access* **2019**, *7*, 110439–110450. https://doi.org/10.1109/access.2019.2934147.
10. Yang, Y.; Wei, L.; Wu, J. Block-smpc: A blockchain-based secure multi-party computation for privacy-protected data sharing. In Proceedings of the 2020 2nd International Conference on Blockchain Technology 2020, Hilo, HI, USA, 12–14 March 2020; pp. 46–51. https://doi.org/10.1145/3390566.3391664.
11. Liu, F.; Yang, J.; Kong, D.; Qi, J. A Secure Multi-party Computation Protocol Combines Pederson Commitment with Schnorr Signature for Blockchain. In Proceedings of the 2020 IEEE 20th International Conference on Communication Technology (ICCT), Nanning, China, 28–31 October 2020; pp. 57–63. https://doi.org/10.1109/icct50939.2020.9295819.
12. Liu, F.; Yang, J.; Li, Z.; Qi, J. A secure multi-party computation protocol for universal data privacy protection based on blockchain. *J. Computer. Res. Dev.* **2021**, *58*, 281–290.
13. Che, X.; Zhou, T.; Li, N. Optimization of NTRU multi-key homomorphic encryption scheme. *Eng. Sci. Technol.* **2020**, *52*, 186–193.
14. Che, X.; Zhou, T.; Li, N. Modified multi-key fully homomorphic encryption based on NTRU cryptosystem without key-switching. *Tsinghua Sci. Technol.* **2020**, *25*, 564–578. https://doi.org/10.26599/tst.2019.9010076.
15. Daniel, E.; Tschorsch, F. IPFS and Friends: A Qualitative Comparison of Next Generation Peer-to-Peer Data Networks. *IEEE Commun. Surveys. Tutor.* **2022**, *24*, 31–52. https://doi.org/10.1109/comst.2022.3143147.
16. Hua, W.; Gao, Y.; Lyu, M.; Xie, P. Research on Bloom filter: A survey. *J. Comput. Appl.* **2022**, *42*, 1729–1747.
17. Wan, S.; Li, M.; Liu, G.; Wang, C. Recent advances in consensus protocols for blockchain: A survey. *Wirel. Netw.* **2020**, *26*, 5579–5593. https://doi.org/10.1007/s11276-019-02195-0.
18. Xu, G.; Bai, H.; Xing, J. SG-PBFT: A secure and highly efficient distributed blockchain PBFT consensus algorithm for intelligent Internet of vehicles. *J. Parallel Distrib. Comput.* **2022**, *164*, 1–11. https://doi.org/10.1016/j.jpdc.2022.01.029.
19. Lyubashevsky, V.; Peikert, C.; Regev, O. On ideal lattices and learning with errors over rings. In *Theory and Application of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6110, pp. 1–23. https://doi.org/10.1007/978-3-642-13190-5_1.
20. López-Alt, A.; Tromer, E.; Vaikuntanathan, V. On-the-fly multiparty computation on the cloud via multikey fullyhomomorphic encryption. In Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing, New York, NY, USA, 20–22 May 2012; pp. 1219–1234. https://doi.org/10.1145/2213977.2214086.