information

Article

A Modifiable Blockchain Based on the RE-TNG Node Selection Method

Rongtao Chen [1], Chao Li [2,*], Bingrong Dai [2] and Shaohua Zhang [3]

[1] Faculty of Information, Shanghai Ocean University, Shanghai 201306, China; brett_chen1628@163.com
[2] Shanghai Development Center of Computer Software Technology, Shanghai 201112, China; dbr@sscenter.sh.cn
[3] Shanghai Business School, Shanghai 200235, China; zhangsh@sbs.edu.cn
* Correspondence: lc@sscenter.sh.cn

Abstract: Blockchain technology, characterized by its immutability and decentralization, enables the creation of permanent and tamper-resistant records once data are uploaded, making it widely applicable in scenarios requiring data authenticity and reliability. However, the immutability of on-chain data poses significant security risks, as erroneous or illegal data become difficult to correct or remove once recorded. Editable blockchain technology offers a potential solution for on-chain data modification. Nevertheless, existing approaches face several challenges, including the impact of malicious nodes on the security and efficiency of data modification, excessive centralization in the management of modification rights and trapdoor keys, and cumulative issues in reputation-based traditional node grouping methods. To address these challenges, this study proposes an RE-TNG (Reputation Evaluation-Twice Node Grouping) node selection method and an editable blockchain scheme based on it. The RE-TNG method employs a two-stage grouping process following reputation-based node ranking. The first grouping stage uses a Fibonacci sequence-based rule to mitigate the issue of cumulative reputation values over time. The second grouping stage selects high-reputation nodes within groups to ensure the trustworthiness of selected nodes. Trapdoor keys are collaboratively generated by the high-reputation node group, achieving decentralized trapdoor management. Modification nodes are randomly chosen from the high-reputation group, ensuring both integrity and decentralization in modification authority. Comparative analyses and experimental evaluations against traditional random node selection and grouping methods demonstrate the feasibility of the proposed scheme, showcasing a superior performance in terms of security and modification efficiency.

Keywords: editable blockchain; chameleon hash function; verifiable secret sharing; group selection

Academic Editor: Nelly Leligou

Received: 23 December 2024
Revised: 14 January 2025
Accepted: 20 January 2025
Published: 23 January 2025

Citation: Chen, R.; Li, C.; Dai, B.; Zhang, S. A Modifiable Blockchain Based on the RE-TNG Node Selection Method. *Information* **2025**, *16*, 83. https://doi.org/10.3390/info16020083

1. Introduction

Blockchain is a decentralized distributed ledger technology that stores data in blocks and connects them in a chain-like structure, providing a transparent, trustworthy, and secure method of data storage and transmission. Each block contains a set of transaction records and the hash value of the previous block, ensuring the integrity and coherence of the entire chain. Data security is the fundamental goal of blockchain. As a decentralized storage system, blockchain stores vast amounts of critical data, including transactions, user information, smart contract code, and intermediate states of execution. These data are vital and form the primary target for blockchain security [1–3].

Decentralization and immutability have long been regarded as the unshakable foundational characteristics of blockchain. However, the concept of editable blockchain has emerged as a controversial research topic in recent years [4]. The "editable" feature introduces a centralized editing authority alongside the decentralized ledger system, potentially creating a "loophole" where data, even after consensus and being added to the chain, could face centralization or malicious tampering. While this concern is valid, practical blockchain applications reveal pressing demands for data editing in areas such as information regulation, privacy protection, data updates, and scalability. Addressing the security challenges introduced by the centralized editing authority is thus a critical focus for current editable blockchain solutions [5].

Most existing editable blockchain schemes utilize chameleon hash functions, a concept introduced by Krawczyk et al. in 1998 [6]. The defining feature of chameleon hash functions is that, with the possession of trapdoor information, one can modify the hash input without altering the hash output. In 2017, Ateniese et al. [7] proposed a research scheme for editable blockchain based on chameleon hash functions. The core idea was to replace the internal hash function of the blockchain with a chameleon hash function, breaking the second pre-image resistance of standard hashes. By using the trapdoor information, arbitrary modifications to the block data can be achieved without changing the hash output. In recent years, research on achieving editability while ensuring security and data reliability has made further progress. Li Peili et al. [8] improved upon the work in [7] by incorporating secret sharing techniques. They distributed the trapdoor information of each node to other nodes and enabled block ledger modifications only after a modifier node was selected via consensus voting to recover the trapdoor. Since the modifier node is selected using a random function, direct interactions between nodes are avoided, improving both the efficiency of corrections and the degree of decentralization. However, both [7] and [8] rely on a single node for trapdoor key generation and distribution, which leads to excessive centralization and compromises system security, running counter to the decentralized nature of blockchain. Furthermore, if non-trustworthy nodes are present among the modifier nodes, system security is at risk. Ashritha et al. [9] replaced the secret sharing techniques used in [7] and [10] with non-linear secret sharing functions, further enhancing security. In 2021, Gao Wei et al. [11] proposed the concept of a one-time chameleon hash function, which enables a revisable blockchain where each block is allowed to be modified at most once. Any subsequent modification to a block would result in a penalty leading to the collapse of the blockchain. However, these approaches still depends on a single trusted node. Gu Kang et al. [12] introduced a supervisor group and a reputation-based evaluation mechanism to determine modifier nodes based on reputation scores. However, as the number of modification requests increases, reputation accumulation can lead to the concentration of editing authority in a few nodes, thus continuing to face the issue of excessive centralization.

In summary, existing editable blockchain solutions based on chameleon hash functions rely on modifier nodes obtaining trapdoor keys to modify on-chain data. The security of these schemes depends on the trustworthiness of both the modifier nodes and the nodes responsible for generating and distributing the trapdoor keys. This reliance results in a high degree of centralization, contradicting the foundational principle of decentralization in blockchain systems [13]. Moreover, due to the presence of dishonest or inactive nodes, there is considerable room for improvement in both modification efficiency and system security. Current reputation-based node selection methods further exacerbate the issue by allowing reputation values to accumulate disproportionately over time, leading to a concentration of authority in a few nodes. This trend toward centralization undermines the decentralized nature of blockchain.

Therefore, this paper addresses the issues in existing modifiable blockchain schemes, including the impact of malicious nodes on modification security and the excessive centralization in node selection and trapdoor management. To tackle these challenges, we propose a Reputation Evaluation-Twice Node Grouping (RE-TNG) node selection method and design an improved blockchain modification scheme based on this approach. The main contributions of this paper are as follows:

Designing a Reputation Evaluation Reward-Penalty Model: This model aims to enhance the active participation of nodes in the modification process while increasing the cost of malicious behavior for dishonest nodes. By doing so, it reduces the negative impact of both malicious and inactive nodes on modification efficiency and overall system security.

1. Proposing the RE-TNG Node Selection Method: This method involves a two-stage grouping process based on reputation values. Initially, nodes are ranked by their reputation values, and the Fibonacci function is applied to perform the first grouping. Within each group, nodes are further sorted in descending order of reputation. Subsequently, a second grouping is conducted to select candidate nodes for modification and trapdoor management from the initial groups.
2. The Fibonacci-based first grouping addresses the issue of reputation value accumulation prevalent in traditional reputation-based algorithms. The second grouping resolves the problem of randomly selecting modifier nodes, which often fails to ensure trustworthiness. Additionally, this method improves upon conventional modifiable blockchain schemes by limiting the voting and verification processes to nodes with high reputation scores. This reduces the influence of malicious and inactive nodes on modification security and efficiency.
3. Enhancing Trapdoor Key Management with Multi-Party Collaboration: The proposed approach replaces the reliance on a single node for generating trapdoor keys in existing schemes with a multi-party collaborative key generation mechanism. Furthermore, the trapdoor recovery and redistribution processes are facilitated through secret resharing techniques. This improvement addresses the centralization issue inherent in traditional secret sharing methods that depend on authoritative nodes for trapdoor key generation and distribution.

2. Related Knowledge

2.1. Chameleon Hash

Hash Functions: The hash chain generated by hash functions provides blockchain systems with tamper-resistance properties. A hash function is a mathematical function that maps input data of arbitrary size to a fixed-length output string. Its general formula can be expressed as follows: $h = H(m)$, where h is the output of the hash function, also known as the hash value; H represents the hash function; and m is the input message or data.

Hash functions have the following two properties:

- **Collision Resistance**: It is computationally infeasible to find two distinct inputs m_1 and m_2 such that $H(m_1) = H(m_2)$.
- **High Sensitivity**: Even a small modification in the input data m, such as a single bit change, will cause a significant change in the output $H(m)$, resulting in a drastically different hash value.

The above two properties are key reasons for the tamper-resistance of blockchain systems. The chameleon hash function, proposed by Krawczyk et al. [14] as part of a new digital signature scheme called "chameleon signatures", differs from traditional hash functions. In addition to possessing the collision resistance and high-sensitivity properties of traditional hash functions, it introduces the concept of a "trapdoor key." The trapdoor key can be artificially set, and anyone who possesses it can easily find a collision, allowing them to modify the input data arbitrarily without changing the hash output. For users without the trapdoor key, the chameleon hash function still maintains collision resistance.

Formally, a chameleon hash function can be defined as follows: Given a trapdoor key tk, for any given data and arbitrary parameter pair, it is possible to ensure that $CH(x', r') = CH(x, r)$, where x represents the original data on the blockchain, and x' represents the modified data, with $x \neq x'$.

Formally, a chameleon hash function consists of four algorithms, as shown in Equation (1):

$$CH = \{GenKey, CHash, CHVerify, CHCol\} \tag{1}$$

(1) $GenKey : GenKey(1^k) \rightarrow (hk, tk), k \in N$. The key generation algorithm for chameleon hash functions: Given a security parameter $k \in N$, it outputs a public key hk and a private key tk (where tk is typically referred to as the trapdoor).

(2) $CHash : CHash(hk, x, r) \rightarrow h$. The chameleon hash generation algorithm: Given the chameleon hash public key hk, arbitrary data x, and a random value r, the output is the hash value h.

(3) $CHVerify(hk, x, h) = D$. The chameleon hash verification algorithm: Given the public key hk, arbitrary data x, and the hash value h, it outputs a verification result D. If the hash value h is correct, D = 1 otherwise, D = 0.

(4) $CHCol : CHCol(tk, (x, r), x') \rightarrow r'$. The hash collision discovery algorithm: Given the trapdoor tk, the tuple (x, r) consisting of the original data x and the random value r, along with the modified data x', it outputs a new random value r'. To ensure that:

$$h = CHash(hk, x, r) = CHash(hk, x', r')$$

The value of h satisfies:

$$CHVerify(hk, x, h) = CHVerify(hk, x', h) = 1$$

In summary, as long as the trapdoor key tk is available, one can modify the data on the blockchain such that $CH(x', r') = CH(x, r)$. In other words, possessing the trapdoor key means having the authority to modify the blockchain system. Therefore, the management of the trapdoor key becomes even more crucial.

2.2. Verifiable Secret Sharing (VSS)

Secret sharing is a cryptographic protocol proposed by Shamir et al. [15], which allows a secret to be divided into several shares and jointly held by multiple participants. Each participant only obtains a portion of the secret information and cannot independently access the complete secret. That is, the information held by each individual member does not reveal any part of the complete secret. Only when specific conditions are met can the participants reconstruct the original secret information.

In reference [15], the secret is divided into n shares, and at least c shares (where $c > \frac{n}{2}$) are required to reconstruct the complete secret. Chor et al. [16] proposed verifiable secret sharing (VSS) technology based on secret sharing. VSS introduces a verification algorithm, allowing participants to verify whether the secret shares they hold are correct and complete. This ensures that participants are not deceived by dishonest members providing false shares, which could otherwise prevent the successful reconstruction of the original secret.

The verifiable secret sharing (VSS) scheme satisfies two key properties: verifiability and unpredictability. VSS consists of three components: the secret sharing algorithm; the share verification algorithm, $Verify$; and the secret reconstruction algorithm, $Recover$. These are defined as follows:

Given a secret s and a set of participants $p = \{p_1, p_2, \ldots, p_n\}$ involved in the sharing process, the scheme proceeds as follows:

(1) $Share$. Secret Sharing Algorithm: The secret s is divided into n shares s_i, with each share distributed to a participant p_i through a secure channel. The secret information is split into the coefficients of multiple polynomials, and participants can also compute the commitment values of their respective shares s_i and make them public within the system.

(1) $Verify$. Share Verification Algorithm: When there is a need to verify the consistency of the shares, participants p_i can exchange shares or information with each other. Using a verification algorithm (such as digital signatures or zero-knowledge proofs), the received shares are validated to ensure their correctness. Participants p_i verify each other's shares and ensure that they have not been tampered with or forged.

(3) $Recover$. Secret Reconstruction Algorithm: To recover the complete secret information, a threshold number of correct c secret shares must be collected. After each participant $p_1, p_2, \ldots, p_c$ involved in the cooperative secret recovery validates the secret's validity using the share verification algorithm, the secret can be reconstructed using a secret reconstruction method.

2.3. Verifiable Random Functions (VRFs)

The verifiable random function (VRF) is a cryptographic function proposed by Micali et al. It is a random number generator that produces unpredictable random numbers. Anyone can verify the validity of the random number generated by the VRF by checking the proof and verifying whether the hash computation is correct. Performing the hash operation requires the VRF private key, but anyone with the public key can verify the result of the hash computation. VRF has two main properties: verifiability and randomness. The VRF includes four algorithms:

(1) Key generation algorithm, VRF_{Gen}.
(2) Random number generation algorithm, VRF_{val}.
(3) Proof generation algorithm, VRF_{Proof}.
(4) Random number verification algorithm, VRF_{Verify}.

3. RE-TNG Node Selection Method

The scheme proposed in [7] involves all nodes in the modification process, leading to excessive communication resource consumption, low modification efficiency, and significant efficiency degradation when there are numerous inactive or dishonest nodes. However, the research in [12,17–19] reveals that simple node selection methods result in higher-reputation nodes being more likely to be chosen over time. This feedback loop further increases their reputation, ultimately concentrating reputation values within a few top nodes in the system. Consequently, traditional reputation-based grouping schemes suffer from reputation value accumulation, leading to severe centralization issues in the later stages of the system.

To address this, drawing on the concept of consensus node selection from [17], this paper proposes an improvement by designing the RE-TNG node selection method, which integrates the proposed node reputation reward and penalty model. It introduces the concepts of modification node candidate groups and trapdoor management node groups. This approach mitigates the reputation value accumulation issue, ensuring long-term system stability and decentralization. Additionally, the method guarantees the integrity of the selected nodes, thereby safeguarding the security of trapdoor generation management and the assignment of modification rights.

3.1. Node Reputation Reward and Penalty Model

To reduce the impact of dishonest and inactive nodes on the modification process while encouraging greater participation in the final election of modification nodes, a node reputation reward and penalty model is proposed. The core principle of this model is to quantitatively assess node behavior and apply corresponding rewards or penalties based on the reputation score. This approach motivates nodes to engage more actively in the blockchain modification process while effectively curbing malicious actions by dishonest nodes. Most reputation evaluation models in existing blockchain systems rate nodes based on their behavior during the consensus process [20]. This method builds upon the reputation reward and penalty mechanism described in [17] and the dynamic reputation evaluation model outlined in [21]. Specifically, it adapts the dynamic reputation evaluation model from [21], which assesses consensus nodes based on their roles and behaviors during the consensus process, to evaluate nodes' roles and actions in the modification process. By integrating the reward and penalty mechanism from [17], the model establishes a reputation evaluation and reward system tailored to the proposed approach. Reputation thresholds are defined to categorize nodes into different states, with varying degrees of rewards and penalties applied based on their state. This enables more timely evaluation and feedback on node behavior throughout the modification process. The detailed content of the node reputation reward and penalty model employed in this method is presented below.

After initializing the blockchain system, the initial reputation score for each node is set to 60, with the maximum reputation score capped at 100. Once a node's reputation score reaches 100, it will no longer increase. Five reputation levels are defined, as outlined in Table 1.

Table 1. Credit value level table.

Level Symbols	Level Scores	Evaluation
R_{good}	90	Excellent Reputation Score
R_{better}	80	High Reputation Score
R_{normal}	70	Normal Reputation Score
R_{init}	60	Initial Reputation Score
$R_{\min}$	50	Low Reputation Score

If the reputation score of a node $R_i < R_{\min}$, that node will no longer be allowed to participate in the blockchain modification process. The node's reputation score directly influences its permissions and status within the blockchain system.

Based on the different reputation levels, roles, and whether a node engages in malicious behavior during the modification process, reputation rewards and penalties are applied. If a node does not engage in malicious behavior or show passive conduct during the modification process, its reputation score will be rewarded according to Formula (2); conversely, if a node engages in malicious behavior or exhibits passive conduct, its reputation score will be penalized according to Formula (3).

Reputation Reward Formula:

$$R_i = \begin{cases} R_i^p + 0.1 \times C, R_i^p \geq R_{good} \\ R_i^p + 0.3 \times C, R_{better} \leq R_i^p < R_{good} \\ R_i^p + 0.5 \times C, R_{normal} \leq R_i^p < R_{better} \\ R_i^p + 0.7 \times C, R_{init} \leq R_i^p < R_{normal} \\ R_i^p + 0.9 \times C, R_{min} \leq R_i^p < R_{init} \\ 0, R_i^p < R_{min} \end{cases} \tag{2}$$

Penalty Formula for Reputation Value:

$$R_i = \begin{cases} R_i^p - V_1 \times C, R_i^p \geq R_{good} \\ R_i^p - V_2 \times C, R_{init} \leq R_i^p < R_{good} \\ R_i^p + (1 + \frac{t}{T}) \times C \times v, R_{min} \leq R_i^p < R_{init} \\ 0, R_i^p < R_{min} \end{cases} \quad (3)$$

In Equations (2) and (3) mentioned above, R_i represents the reputation value of a node after the current reward or penalty process. R_i^p denotes the reputation value of the node from its last participation in the modification process. The system evaluates the node's behavior during the modification process to determine whether it acted positively or with integrity (the evaluation method is detailed in Section 4.1.4. Based on the assessment, the reputation value of the node is adjusted using Equation (2) or Equation (3).

V_1 and V_2 denote the penalty intensity coefficients, which are configured according to different system environments. In this model, $V_1 = 15$ and $V_2 = 10$. For nodes with reputation values within the ($R_{\min}$, R_{init}) range, participation in the modification process is restricted for a specific period, as expressed in Equation (3). Here, t represents the number of days, starting at an initial value of 1 and increasing progressively with time. T is a fixed constant that indicates the restriction period in days. v denotes the growth rate, and both v and T can be adjusted based on different application scenarios. In this model, $T = 10$ and $v = 1$. Nodes with reputation values below $R_{\min}$ are prohibited from participating in blockchain modifications. C is the identity coefficient, which takes different values depending on the identity status of the node during its last modification process. The specific values for R_i are provided in Tables 2 and 3.

Table 2. The value of c in the reward formula: Formula (2).

Identity in the Previous Modification Process	Values
Nodes Not Selected for the Trapdoor Management Group and Modifier Candidate Group	1
Members of the Trapdoor Management Group	1.1
Members of the Modifier Candidate Group	1.2
Final Editors	1.3

Table 3. The value of c in the penalty formula: Formula (3).

Identity in the Previous Modification Process	Values
Inactive Nodes (Passive Nodes)	0.9
Dishonest Nodes	1.3

By introducing a reputation model, the identity and behavior of nodes are linked to their reputation, thereby increasing their motivation to participate in the modification process. This model encourages nodes to engage in more positive behaviors, raises the cost for dishonest nodes to engage in malicious activities, and penalizes inactive or passive nodes. As a result, the system motivates nodes to actively participate in the modification process, improving modification efficiency. Overall, this approach reduces the negative impact of inactive and dishonest nodes on modification efficiency. Additionally, by rewarding nodes with reputation value based on different reputation levels using specific reward formulas, the reputation growth of nodes is effectively stabilized. Specifically, high-reputation nodes receive relatively smaller rewards, while low-reputation nodes receive greater rewards. This design incentivizes low-reputation nodes to participate more actively in the modification process and demonstrate positive behaviors in order to earn reputation rewards.

Consequently, even in the later stages of the system, reputation values are not concentrated in a few nodes, ensuring a more decentralized maintenance of reputation values across the entire system.

3.2. RE-TNG Node Selection Method Design

In the RE-TNG node selection method, let the total number of nodes participating in the modification process be $N(N \geq 5)$, and the number of nodes with a reputation value greater than or equal to R_{init} be $N_r(N_r \geq 5)$. The nodes are grouped according to the Fibonacci function rule, where $Node_i \in \{Group_1, Group_2, \ldots, Group_k\}$ represents the group to which a certain node is assigned, and $Group_k$ represents the last group. This grouping rule satisfies the recursive nature of the Fibonacci function $(n \geq 3, n \in N^*)$, as shown in Equation (4):

$$\begin{cases} S(1) = S(2) = 1 \\ S(n) = S(n-1) + S(n-2) \end{cases} \tag{4}$$

All nodes with a reputation value greater than or equal to R_{init} that participate in the modification process are grouped into k groups based on the Fibonacci function rule. The group $Group_{k-1}$ contains $S(k-1)$ nodes, where $Node_{ij}$ represents the number of nodes in the group $Group_i$, and j indexes the nodes within the group. Within the group $Group_k$, the number of nodes is $N_r - \sum_{a=1}^{k-1} S(a)$ when $i = k$ nodes are selected. The node hierarchy is illustrated in Figure 1. Figure 1 illustrates that the nodes are divided into k groups. The first and second groups each contain only one node, while the third group consists of two nodes. The number of nodes in the subsequent groups is allocated according to the Fibonacci function rule.

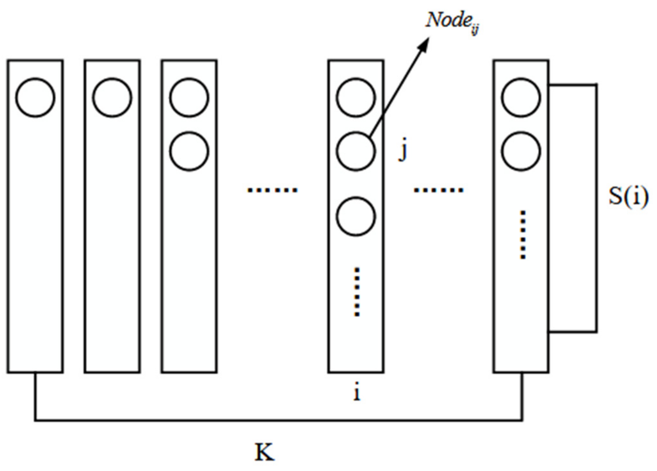


Figure 1. A hierarchical diagram of node grouping based on the Fibonacci sequence.

After grouping according to the Fibonacci sequence rule, the nodes within each group remain sorted in descending order of reputation due to sequential selection. The first node in each group, such as $Node_{i1}$ in $Group_i$, is the node with the highest reputation in that group. $Node_{i1}$ is then added to the modifier node candidate group, meaning the top-reputation node in each group is defined as a member of the modifier candidate group.

For groups other than the first two (i.e., $Group_1$ and $Group_2$), the second node in each group, such as $Node_{i2}$ in $Group_2$, is the node with the second-highest reputation in that group. $Node_{i2}$ is added to the trapdoor management node group, meaning the second-reputation node in each group, except for the first two groups, is defined as a member of the trapdoor management group.

The grouping rules are illustrated in Figure 2. The specific roles of the modifier candidate group and the trapdoor management node group will be detailed in Section 4.1.

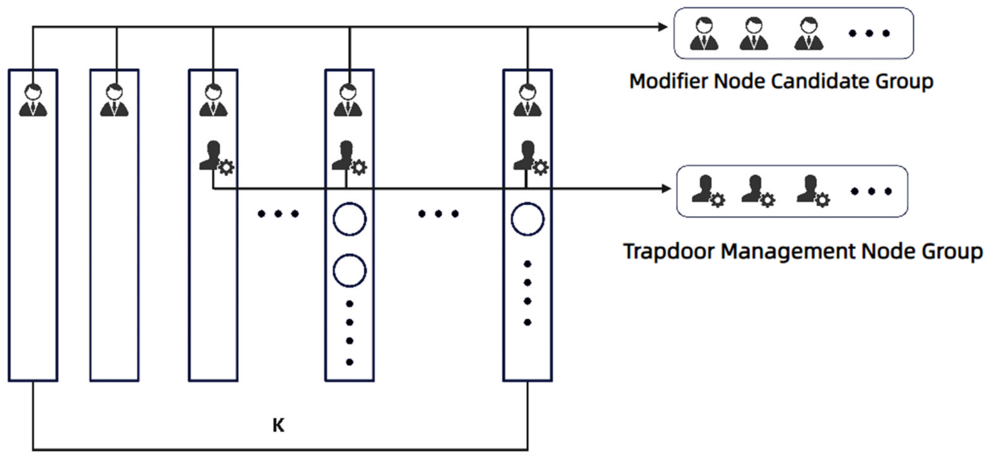


Figure 2. Selection rules for modification node management group and trapdoor management node group.

According to the above rules, nodes participating in the modification process with a reputation value greater than or equal to R_{init} are ranked based on their reputation values. These nodes are then grouped in the first grouping phase according to the Fibonacci function rule, sequentially filling each group. From these groups, members of the modification node candidate group and trapdoor management node group are selected. Since the reputation values of nodes within each group are relatively similar, and nodes from both high-average-reputation groups and low-average-reputation groups are included in the modification node candidate group, the probability of becoming a final modification node is equal for all.

Compared with the traditional reputation-based grouping and selection algorithms used in [12,17,18], this method addresses the issue of reputation value accumulation, thus avoiding system centralization in later stages and increasing the incentive for nodes to participate in the modification process. Furthermore, the second grouping phase allows only a subset of nodes to participate in the final selection of modification nodes, reducing the excessive communication resource consumption and low modification efficiency caused by the full participation of all nodes in traditional schemes.

4. Design of a Modifiable Blockchain Scheme Based on the RE-TNG Node Selection Method

4.1. Scheme Definition

To address the issues in current modifiable blockchain schemes—where dishonest and inactive nodes negatively impact modification efficiency, and the excessive centralization of modification authority and trapdoor management undermines the system—a modifiable blockchain scheme based on the RE-TNG node selection method is proposed. This scheme leverages reputation-based grouping and decentralized decision-making to enhance efficiency and fairness in the modification process. The structure and workflow of the proposed scheme are illustrated in Figure 3.

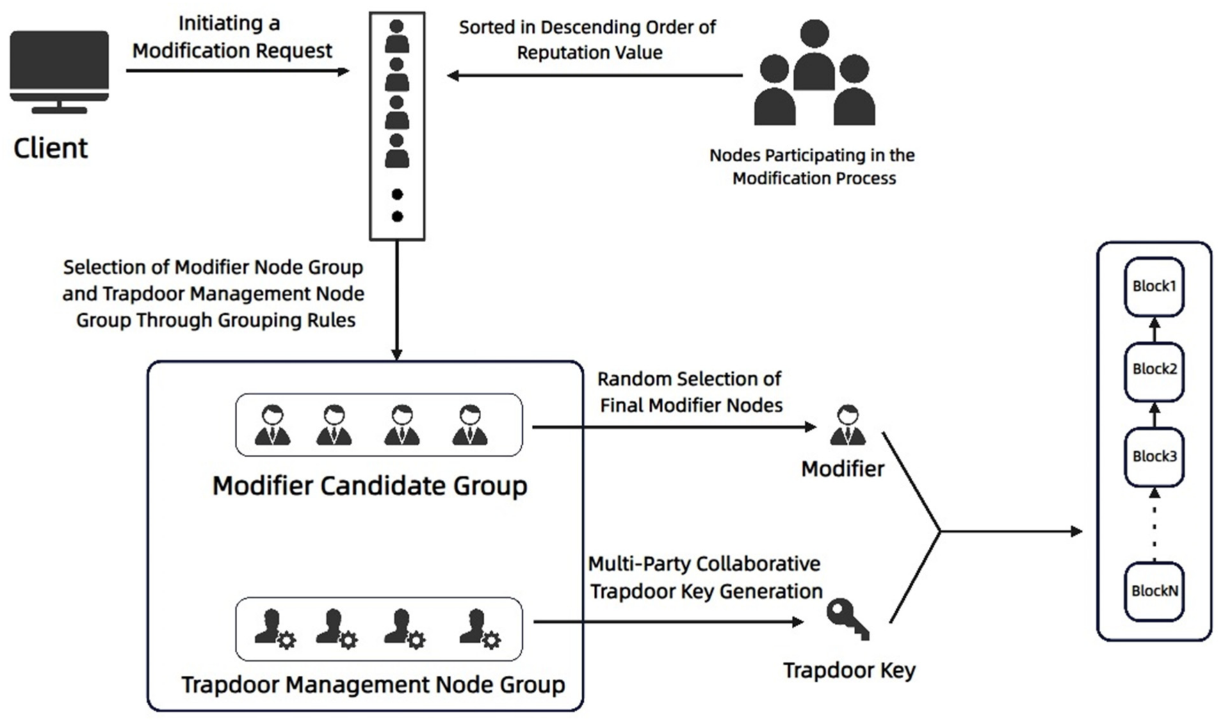

Client
Initiating a Modification Request
Sorted in Descending Order of Reputation Value
Nodes Participating in the Modification Process
Selection of Modifier Node Group and Trapdoor Management Node Group Through Grouping Rules
Modifier Candidate Group
Trapdoor Management Node Group
Random Selection of Final Modifier Nodes
Modifier
Multi-Party Collaborative Trapdoor Key Generation
Trapdoor Key
Block1
Block2
Block3
BlockN

Figure 3. The editable blockchain model based on the RE-TNG node selection method.

The modifiable blockchain scheme based on the RE-TNG node selection method consists of six stages: initialization stage; RE-TNG node selection method—selection stage; final modifier node selection stage; voting stage; trapdoor generation stage; data modification and verification stage (the following process needs to be read and understood in conjunction with Figure 3, which is a modifiable blockchain model based on the RE-TNG node selection method). The following will provide a detailed description of the six stages of the process, with the variables and definitions provided in Table 4. The roles involved in this scheme are as follows:

1. Modifier Node Candidate Group: In the blockchain modification process, the nodes participating in the modification are first sorted based on their reputation scores. After sorting, they are grouped according to the Fibonacci function rule. The first node in each group is selected to join the modifier node candidate group. Then, the final modifier nodes are chosen through a verifiable random function (VRF)**,** ensuring the selection process is both random and verifiable. This method ensures fairness, transparency, and prevents any manipulation in the selection of the nodes that will execute the blockchain modifications.
2. Trapdoor Management Node Group: After the nodes participating in the modification process are sorted based on their reputation scores, they are grouped using the Fibonacci function rule. The second node in each group is selected to join the trapdoor management node group. This group is responsible for two main tasks: collaborative trapdoor generation and voting on the final modifier node selection.

Table 4. Variable symbols and definitions.

Variable Symbols	Definitions
N_{start}	Modification Request Initiator Node
AD_{start}	User Address of the Modification Request Initiator Node
AD_{update}	User Address of the Final Modifier Node
N_{update}	Final Modifier Node
N_i	A Node Participating in the Process
N_c	Modifier Node Candidate Group Members
N_{tk}	Trapdoor Management Node Group Members
$List_{init}$	Initial Request List
$List_{vote}$	Voting Request List
$Group_{init}$	Nodes Group Sorted by Reputation in Descending Order
λ	VRFval Random Number Generated by $VRFval$ Algorithm
b	Voting Opinion
$poll$	Number of Agreeing Votes
$Node_{i1}$	First Node of the i-th Group
$Node_{i2}$	Second Node of the i-th Group
$Proof$	$\text{VRF}_{\text{Proof}}$ Proof Generated by the VRF_{Proof} Algorithm
$result$	$\text{VRF}_{\text{Verify}}$ Result Set Generated by the VRF_{Verify} Algorithm
PK_{VRF}	VRF Public Key Generated by the VRF_{Gen} Algorithm
SK_{VRF}	VRF Private Key Generated by the VRF_{Gen} Algorithm

4.1.1. Initialization Stage

At a certain moment, N_{start}, a particular node in the blockchain system requests to modify the transaction data in a specific block, changing x to x'. The node N_{start} will create an initial request list, $List_{init}$, consisting of the original data x, the target data to be modified x', a random number r, and its own account address AD_{start}, forming a tuple (x, x', r', AD_{start}). This request list will be broadcast to the other nodes in the blockchain system. After receiving the request list, the other nodes will decide whether to participate in the modification process. Once the participating nodes are determined, they are sorted in descending order by their reputation, forming a group $Group_{init}$.

4.1.2. RE-TNG Node Selection Method—Selection Stage

For all nodes in $Group_{init}$ with a reputation value greater than or equal to R_{init}, the nodes are grouped according to the Fibonacci function rule, resulting in $\{Group_1, Group_2, \ldots, Group_k\}$. All $Node_{i1}$ nodes in $\{Group_1, Group_2, \ldots, Group_k\}$ are selected as members of the modification node candidate group N_c, and all $Node_{i2}$ nodes are selected as members of the trapdoor management node group N_{tk}. A detailed explanation of the RE-TNG node selection method is provided in Section 3.2.

4.1.3. Final Modifier Node Selection Stage

The final modification node N_{update} is selected from the modification node candidate group members N_c. To ensure the security of the method, the node responsible for modifying blockchain data must be unpredictable and possess a degree of randomness. Therefore, this paper employs a verifiable random function (VRF). By determining the number of members in the modification node candidate group, a range is assigned to each node. The VRF is then used to generate the required PK_{VRF} (public key) and SK_{VRF} (private key) for the next step [22]. Using the VRF random number generation algorithm $VRF_{val}(SK_{VRF}, x') \rightarrow \lambda$, the private key SK_{VRF} and the target data to be modified, x', are input to produce a random number λ. The generated random number λ is mapped to the range assigned to the members of the modification node candidate group. Based on this mapping, the member node corresponding to the interval is selected, thereby achieving randomness in node selection. The selected final modification node N_{update} must then compute a zero-knowledge proof (PROOF): $VRF_{Proof}(SK_{VRF}, x') \to Proof$. The computed proof is broadcast to all nodes in the blockchain for later validation and accountability.

4.1.4. Voting Stage

The selected final modification node N_{update} combines the original data x to be modified, the modified data x', and the user address AD_{update} of the final modification node N_{update} into a triplet voting request list $List_{vote} \to (x, x', AD_{update})$, which is then broadcast to all members of the trapdoor management node group. Upon receiving the voting request list, all N_{tk} members vote on whether to approve N_{update} modifying data x to x'. This stage includes one algorithm:

Voting Algorithm: $Vote\Big(\big(x, x', AD_{update}\big), (b_0, b_1, \ldots, b_n)\Big) \to poll$. Input: Voting request list $List_{vote}$. Voting opinions b from all N_{tk} members, where b = 1: Agree; b = 2: Disagree; b = 0: Abstain. Output: The number of agreeing votes.

If the number of agreeing votes $poll > \frac{w}{2}$ (where w is the number of participating trapdoor management node group members, N_{tk}), then N_{update} is confirmed as the final modification node authorized to modify the blockchain data. If the number of agreeing votes $poll < \frac{w}{2}$, the final modifier node selection phase is repeated to select another N_{update}, and the voting phase is conducted again. After the voting phase, any N_{tk} members who did not participate or abstained from voting will be classified as inactive nodes. These nodes will have their reputation values reduced during the post-modification reputation evaluation and reward–punishment phase. The detailed reward and punishment mechanism has been elaborated in Section 3.1 of this paper.

4.1.5. Trapdoor Generation Stage

This phase is primarily carried out by the trapdoor management node group members N_{tk}, who are responsible for generating public parameters, digital signatures, and the key pairs (public and private) for the chameleon hash function. In this scheme, the method for generating the trapdoor tk for the chameleon hash is improved from single-node generation to a multi-node collaborative trapdoor generation algorithm adapted to this scheme.

Assuming the current number of trapdoor management node group members is $n(n \geq 5)$, this stage includes three algorithms (Algorithms 1–3).

Algorithm 1: Initialization algorithm

```
$DH\_PGen(\xi) \to P_p$
**Input**: Secure random parameter $\xi$.
**Output**: A public parameter $P_p$.
**Step 1**: Generate a secure random parameter $\xi$.
**Step 2:** Use λ to compute the public parameter $P_p$ according to the predefined cryptographic function.
**Step 3:** Output $P_p$.
```

Algorithm 2: Chameleon hash public and private key collaborative generation algorithm

```
$DS\_Gen(P_p) \to (hk, tk_0, tk_1, \ldots, tk_{n-1})$
**Input**: Public parameter $P_p$.
**Output**: Public key $hk$. Private key (trapdoor) $(tk_0, tk_1, \ldots, tk_{n-1})$, collaboratively managed by $n$ trapdoor management node group members $N_{tk}$.
```

Step 1: **Public key generation.**

1. The public key hk is generated using the public parameter P_p. This key is shared among all nodes and is the same for the entire system, meaning there is only one hk for the whole system.

Step 2: Private key (trapdoor) generation.

1. Each member of the trapdoor management node group N_{tk} generates a partial private key tk_{i} using the public parameter P_p. These partial private keys are then sec urely combined through a multi − party computation protocol to form the complete trapdoor tk.

Step 3: Public key disclosure.

1. The final public key hk is publicly disclosed in the blockchain system, available for use in chameleon hashing operations.

Step 4: Trapdoor Management.

1. The private key (trapdoor) tk is securely distributed and collaboratively managed by the **n** members of the trapdoor management node group N_{tk}. Each member holds a share of the private key, and it requires cooperation from multiple members to access the complete trapdoor tk.

Step 5: Output

1. hk: Publicly disclosed and available for use in the blockchain system.
2. tk: Trapdoor private key, distributed among the **n** members of the trapdoor management node group. Each member holds a share of the private key.

Algorithm 3: Trapdoor re-sharing algorithm

$ReShare(tk_0, tk_1, \ldots, tk_{n-1}) \rightarrow (tk_0, tk_1, \ldots, tk_{t-1})$

Input:

1. n trapdoor shares : $(tk_0, tk_1, \ldots, tk_{n-1})$.
2. Threshold value t.

Output: t trapdoor shares that can be distributed to trapdoor management node group members N_{tk}.

Step 1: Secret sharing.

1. The n trapdoor shares are distributed to the n trapdoor management node group members N_{tk}. The system uses a **verifiable secret sharing** technique to ensure that each member holds a share of the trapdoor private key tk.

Step 2: Threshold.

1. The threshold t is set such that at least **t** trapdoor shares are required to reconstruct the complete trapdoor tk. In this scheme, $t = \frac{w}{2}$, where w is the total number of members in the trapdoor management node group.

Step 3: Trapdoor reconstruction.

1. To recover the full trapdoor tk, more than **t** shares must be obtained. This is ensured by the condition that the number of votes agreeing with the modification, $poll$, must exceed $\frac{w}{2}$, meaning the number of agreeing votes (trapdoor shares) is greater than the threshold t.

Step 4: Share transfer.

1. After the voting process, the trapdoor shares from the agreeing N_{tk} nodes are sent to the final modification node N_{update}.

Step 5: Output.

1. The final trapdoor tk is reconstructed when **t** shares are aggregated, and the resulting trapdoor tk is accessible to the authorized nodes.
2. The agreeing N_{tk} nodes send their respective trapdoor shares to the final modification node N_{update}.

4.1.6. Data Modification Stage

N_{update}, having received the trapdoor key shares $(tk_0, tk_1, \ldots, tk_{n-1})$ from n members N_{tk} and the initial request $List_{init}(x, x', r', AD_{start})$, performs modification operations on the block where the data x in the $List_{init}$ is located. This phase consists of two algorithms (Algorithms 4 and 5).

Algorithm 4: Hash collision detection algorithm

$DH\_Adpt((x, r), x', hk, (tk_0, tk_1, \ldots, tk_{n-1})) \rightarrow r'$

Input:

1. n trapdoor key shares $(tk_0, tk_1, \ldots, tk_{n-1})$.
2. Public key hk.
3. The original data and random number pair (x, r).
4. The modified data x'.

Output: A new random number r'.

Step 1: Compute the random number shares $r_i' = g^e \cdot g^{(x-x')\cdot tk_i}(i \in (1, 2, \ldots, n))$ using the trapdoor shares $(tk_0, tk_1, \ldots, tk_{n-1})$ through collaborative calculation ξ.

Step 2: N_{update} collects the random number shares $r' = \prod_{i-1}^{t} r_i'(i \in (1, 2, \ldots, n))$ to reconstruct r'.

Step 3: Ensure that $DH\_Hash(x, r, hk) =$ $DH\_Hash(x', r', hk)$ holds, where the new random number r' guarantees that the hash values before and after the modification remain consistent.

Algorithm 5: Chameleon hash verification algorithm

$CHVerify(hk, x, h) = CHVerify(hk, x', h) = D = 1$

Input:

1. Modified data x'.
2. New random number r'.

Verification: Check whether the modified data x' satisfies $CHVerify(hk, x, h) = CHVerify(hk, x', h) = D = 1$.

After the modification is completed, the reputation value evaluation and reward–punishment mechanisms are applied to the nodes involved in the modification process using the node reputation reward–punishment model. The detailed mechanisms have been elaborated in Section 3.1 of this paper.

4.1.7. Verification Stage

By referencing the accountability verification method in [23], this scheme designs its own accountability method. Since the final modification node N_{update} in this scheme is selected based on a verifiable random function (VRF), which incorporates a zero-knowledge proof mechanism, other nodes can hold N_{update} accountable during the final modifier node selection phase. Once the nodes obtain the $Proof$, they can verify the modifying node responsible for altering the block using the VRF verification algorithm. The specific algorithm is as follows (Algorithm 6).

Algorithm 6: Verification algorithm

$VRF_{Verify}(PK_{VRF}, x', \lambda, \mathrm{Proof}) \to result$

Input:

1. Public key PK_{VRF} (generated in theVRF_{Gen} phase).
2. Modified data x'.
3. Random number λ.
4. Proof.

Verification: The node to be verified inputs the public key PK_{VRF} generated during the final modification node selection phase VRF_{Gen}, the modified data x', the random number
λ, and the proof $Proof$. The algorithm outputs $result$. Using x' and the result set $result$, the node responsible for modifying the blockchain block is identified, and accountability measures are taken. This method achieves the accountability function of the proposed scheme.

4.2. Decentralized Management of Trapdoor

Existing blockchain modification schemes often rely on traditional chameleon hash functions that depend on a trusted central node or authority. These central entities generate the chameleon hash keys and manage subsequent distribution using verifiable secret sharing (VSS) technology. This approach introduces a high degree of centralization during the initialization phase, conflicting with the decentralized nature of blockchain systems. Moreover, the security of the system is threatened if the distribution node in the secret-sharing scheme is dishonest. Reference [24] introduced a decentralized chameleon hash function to address these issues by decentralizing the trapdoor generation process, avoiding the centralization problems caused by single-node trapdoor management. Building on this, our scheme further improves the decentralized chameleon hash function proposed in [24]. Specifically, we adapt it to support multi-node collaborative key generation within the trapdoor management node group in our system. The proposed scheme employs a multi-party collaborative key generation algorithm to eliminate the reliance on a single-node trapdoor generation approach typical of traditional chameleon hash functions. The trapdoor is generated in a decentralized manner, and nodes participating in collaborative generation hold individual trapdoor shares. To compute a new collision hash random number, the modification node must receive a threshold number of shared random number portions before successfully calculating the correct new random number. This ensures the collision resistance of the hash function, prevents trapdoor leakage, and realizes decentralized trapdoor key management. Below, we detail the four algorithms (Algorithms 7–10) involved in the multi-party collaborative key generation technique.

Algorithm 7: Parameter initialization algorithm

$DH\_PGen(\xi) \to P_p$

Input: Security parameter ξ.

Output: Public parameter P_p.

Algorithm 8: Chameleon hash public and private key collaborative generation algorithm

$DS\_Gen(P_p) \rightarrow (hk, tk_0, tk_1, \ldots, tk_{n-1})$
Input: Public parameter P_p.
Output:

1. A public key hk.
2. n sec ret keys $(tk_0, tk_1, \ldots, tk_{n-1})$, where the n secret keys are held by the n nodes participating in the collaboration.

Step 1: Node N_i selects an $n-1$ degree polynomial, such as,
$f_i(y) = a_0^i + a_1^i y + \ldots + a_{n-1}^i y^{n-1}$
where $a_0^i, a_1^i, \ldots, a_{n-1}^i$ is a random integer and y represents the user identity information of the node. Each node can compute an intermediate value based on this polynomial, resulting in n intermediate values $\left\{(j, f_i(j))|_{j=0}^{n-1}\right\}$ for the polynomial.
Step 2: The n nodes will distribute the i-th
intermediate value (processed as $(f_i(j), g^{f_i(j)}, g^{a_0^j})$ to node j.
Step 3: Node j verifies the correctness of the intermediate values and calculates the output public key $hk = g^s \leftarrow \prod_{j=1}^{n} g^{a_0^j}$ (where s represents the chameleon hash trapdoor key, $s = \sum_{j=0}^{n-1} a_0^j$) and the trapdoor sharing formula $\mathrm{tk}_i = \sum_{j=0}^{n-1} f_i\left(j\right)$. The remaining nodes can obtain the shared portions of the trapdoor key through the above steps.

Algorithm 9: Hash algorithm

$DH\_PGen(\xi) \rightarrow P_p$
Input: Message x, random number r, public key hk (where r is obtained from $r = g^e\left(e \in Z_p\right)$).
Output: Hash value
h (calculated using $h = H(r, x, hk) = r \cdot hk^x = g^{e+s\cdot x}$, where $H : \{0,1\}^* \rightarrow G^*$ is a traditional collision-resistant hash function)

Algorithm 10: Hash collision discovery algorithm

$DH\_Adpt((x, r), x', hk, (tk_0, tk_1, \ldots, tk_{n-1})) \rightarrow r'$
Input:

1. Original data and random number tuple (x, r).
2. Public key hk.
3. Trapdoor key shares tk_i from participating nodes, modified data x'.

Output: New random number r'.
Step 1: Compute trapdoor key share contribution; Node nin_ini
uses the trapdoor sharing formula to compute the random number share
$r_i' = g^e \cdot g^{(x-x')\cdot tk_i}$.
Step 2: Aggregate shares and combine the shares such that $e' = e + (x - x') \cdot tk_i$ is satisfied.
Step 3: Generate a new random number. Derive the new random number r', ensuring $i \in (0, 2, \ldots, n-1)$ holds true.

The verification algorithm of the improved chameleon hash function is consistent with the traditional chameleon hash, so it will not be separately explained.

5. Scheme Analysis

5.1. Security Analysis

Theorem 1. *The scheme possesses editability. For any data x in a given block that need to be modified to x', it is always possible to find a new random number r corresponding to x, such that $DH\_Hash(x, r, hk) = DH\_Hash(x', r', hk)$, and ensure that it passes the validation in the chameleon hash function, $CHVerify(hk, x, h) = CHVerify(hk, x', h) = D = 1$. This guarantees that the modification of data in the block does not alter their output hash value, nor does it affect the hash values of other blocks. Consequently, the integrity, immutability, and security of the entire blockchain system are preserved.*

Proof. In the improved hash collision discovery algorithm, the final modification node only needs to possess more than the threshold t shares of the trapdoor tk to compute the new random number r' using the formula $r_i' = g^e \cdot g^{(x-x')\cdot tk_i}$ $(i \in (1, 2, \ldots, n))$ and $r' = \prod_{i-1}^{t} r_i'(i \in (1, 2, \ldots, n))$. According to the verifiable secret sharing protocol, reconstructing a complete secret requires possession of more than the threshold t secret shares. Through Lagrange interpolation, the complete secret can be reconstructed. In this scheme, given that the public key $hk = g^s$, the formula $h' = r' \cdot hk^{x'} = g^e \cdot g^{(x-x')\cdot s} \cdot g^{sx'} = g^{e+x\cdot s} = r \cdot hk^x = h$ can be used to derive $h' = h$, which can then be utilized to calculate $CHVerify(hk, x, h) = CHVerify(hk, x', h) = D = 1$. As a result, for any random number r corresponding to x, it is always possible to find a new random number r' such that $DH\_Hash(x, r, hk) = DH\_Hash(x', r', hk)$. Thus, it ensures that modifications to the block data maintain consistency in the chameleon hash value, allowing the modified block to pass validation without affecting the hashes of other blocks, thereby preserving the security and integrity of the blockchain. □

Theorem 2. *The RE-TNG node selection method and multi-party trapdoor generation can reduce the impact of malicious nodes.*

Proof. By introducing a reputation evaluation and reward–punishment model, and integrating reputation values into the grouping system, the RE-TNG node selection method effectively filters out well-reputed nodes. This significantly reduces the proportion of malicious nodes participating in the modification process, ensuring the credibility of the entire modification procedure. Over multiple iterations, nodes with reputation values below R_{min} are restricted or prohibited from participating in subsequent modification processes. Consequently, malicious nodes are excluded from selection for both the trapdoor management group N_{tk} and the modification node candidate group N_c, ensuring that these groups remain honest and trustworthy. The multi-party collaborative trapdoor generation mechanism ensures the decentralization of trapdoor management, effectively preventing malicious nodes in traditional schemes from obtaining the trapdoor key and launching attacks on the blockchain system. Only entities holding a number of trapdoor shares exceeding the defined threshold can reconstruct the complete trapdoor information. This guarantees the security of trapdoor management and mitigates the impact of malicious nodes on trapdoor administration. □

Theorem 3. *The reputation reward and punishment model can improve the modification efficiency of the editable blockchain system.*

Proof. The introduction of a reputation reward and punishment model increases the enthusiasm of nodes to participate in the modification process while reducing the impact of inactive nodes on modification efficiency. The mechanism rewards actively participating nodes with an increase in reputation value, which grants them greater operational privileges within the blockchain system. For inactive nodes (e.g., trapdoor management group nodes N_{tk} that abstain from voting during the voting phase), the punishment formula defined in the reputation reward and punishment model deducts their reputation values. After multiple rounds of reputation evaluation, nodes that consistently demonstrate inactivity may lose eligibility to participate in the modification process due to insufficient reputation values. This exclusion prevents their negative behavior from hindering the system's modification efficiency. Therefore, the reputation reward and punishment model not only ensures the security of the entire system but also significantly enhances its modification efficiency. □

5.2. Functional Comparison

This section compares the proposed scheme with existing editable blockchain solutions from four perspectives: decentralized trapdoor management, node selection, accountability, and security. The comparison includes schemes from references [7,8,24] and [12], aiming to comprehensively demonstrate the functional features of the proposed solution.

The solutions in [7,8] rely on traditional verifiable secret sharing (VSS) techniques, where a single central node generates the trapdoor and divides it into multiple secret shares for distribution. However, these approaches exhibit excessive centralization from the outset, contradicting the decentralized nature of blockchain systems. Furthermore, the security of these schemes depends heavily on the integrity of the central node responsible for distributing the secret, making them vulnerable to dishonest behavior from the central node.

Reference [24] addresses this issue by introducing decentralized trapdoor generation, mitigating the centralization problem caused by single-node management. However, it lacks a mechanism to screen nodes involved in the modification process, leading to reduced efficiency and potential risks to security when dishonest nodes participate in the process.

In contrast, the proposed scheme improves upon [24] by incorporating decentralized trapdoor management tailored for a multi-node trapdoor management group. A multi-party collaborative key generation algorithm ensures the decentralization of trapdoor generation, moving away from the traditional single-node approach. Furthermore, the proposed scheme employs a reputation-based reward–punishment model to screen participating nodes, ensuring their reliability and honesty. Nodes with reputation values below a minimum threshold $R_{\min}$ are restricted or prohibited from participating in future modification processes, thus safeguarding the security and efficiency of the system.

Reference [12] introduces a supervisory group and a reputation evaluation model to ensure the honesty of nodes participating in the modification process. However, the management of the supervisory group is cumbersome and impacts the overall efficiency of modifications. Additionally, the reputation evaluation model suffers from reputation accumulation issues, leading to increased centralization over time.

By contrast, the proposed RE-TNG node selection method addresses the problem of reputation accumulation and prevents a scenario where reputation becomes concentrated in a few nodes during repeated modification processes. This avoids the centralization of the system while selecting highly reputable nodes to participate in later processes. As a result, the proposed scheme reduces communication overhead caused by the involvement of all nodes and ensures security without compromising efficiency.

The comparison of specific functions is shown in Table 5.

Table 5. Comparison of editable blockchain schemes.

Scheme	Decentralized Trapdoor Management	Node Selection	Accountability	Security
Reference [7]	No	No	No	Low
Reference [8]	No	No	No	Moderate
Reference [24]	Yes	No	Yes	High
Reference [12]	No	Yes	Yes	High
Proposed Scheme	Yes	Yes	Yes	High

5.3. Experimental Analysis

5.3.1. Experiment Environment

To better validate the various performance aspects of the proposed scheme, simulation experiments were conducted. These experiments include scenarios such as processing multiple block modification requests simultaneously and dealing with varying numbers of malicious nodes disrupting the modification process. The experiments aim to demonstrate the scheme's performance across multiple dimensions. The experimental environment configuration is shown in the table below (Table 6).

Table 6. Environmental configuration (City and country where the device is located: Shanghai, China).

Hardware/Software	Model
Operating System	Windows 11, version 24H2
CPU	12th Gen Intel(R) Core(TM) i5-12400F 2.5 GHz
Memory	KINGBANK 16 G RAM 3200 MHZ

5.3.2. Performance Analysis

1. Success Rate of Correctly Modified Block Data with Different Numbers of Malicious Nodes

This experiment compares the existing solutions, which either do not filter the nodes involved in the modification process or do not implement decentralized gate management, with the proposed solution. The analysis focuses on the success rate of correctly modifying block data under different proportions of malicious nodes in the blockchain system. The solution without node filtering is represented by reference [8], while the solution without decentralized gate management is represented by reference [12]. The results are shown in Figure 4.

From the graph, it is clear that reference [8] follows a more traditional approach to modifiable blockchains, lacking both node filtering for modification participants and decentralized gate management. As a result, when the proportion of malicious nodes increases, the success rate of modification is the lowest among the three solutions. Reference [12] introduces a supervisor group mechanism, which includes a filtering mechanism that improves the system's security to some extent. However, since the management of the gate still relies on trusted nodes, security issues remain significant when malicious nodes increase.

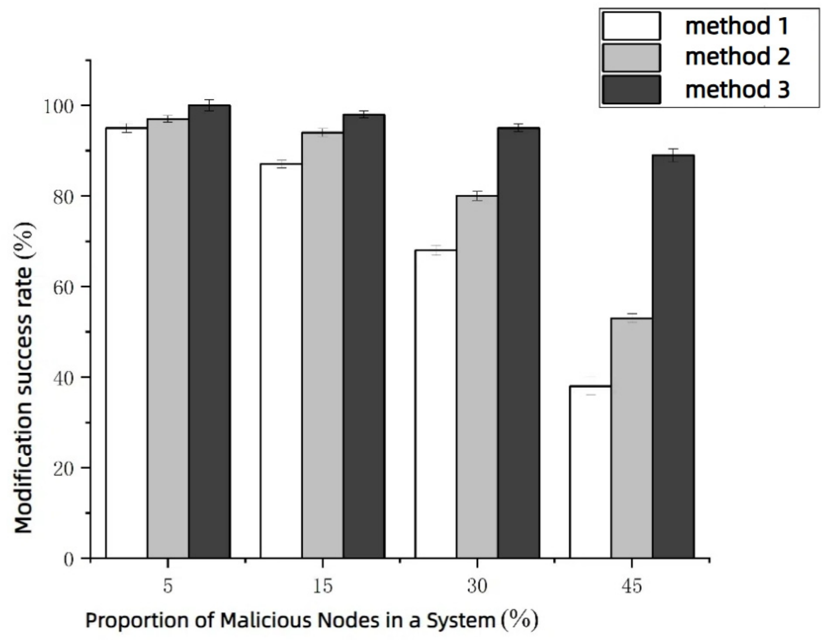


Figure 4. The success rate of correctly modifying block data under different proportions of malicious nodes. (Method 1 references the method in [8], Method 2 references the method in [12], and Method 3 is the method used in this paper).

In contrast, the proposed solution employs the RE-TNG node selection method to filter the candidate modification nodes and gate management nodes, decentralizing both the selection of modification nodes and the management of the gate. This makes the solution more secure. In the experiment, under the influence of various numbers of malicious nodes, the proposed solution consistently demonstrates a higher success rate in modifications compared to the other solutions, indicating superior security.

2. Modification Efficiency when Handling Multiple Modification Requests Simultaneously

This experiment compares the modification efficiency of our proposed scheme with those of [8] and [12] under ideal conditions when multiple modification requests are processed simultaneously, analyzing the time required for each scheme under the same conditions. Reference [8] represents a scheme where modification nodes are selected randomly, while reference [12] introduces a supervisor group for selecting modification nodes. The experimental results are shown in Figure 5.

From the experimental chart, it is observed that when the number of modification requests is less than or equal to 24, the modification efficiency of reference [8] is the best. Since both our scheme and that in [12] require selection and grouping in the initial stage, they both take a certain amount of time initially. However, when the number of modification requests exceeds 24, the scheme given in [8] requires all nodes to participate in the modification process, resulting in significantly higher time requirements than our scheme and that in [12], which only require partial nodes to participate. Therefore, as the number of modification requests increases, both our scheme and that in [12] outperform that from [8] in modification efficiency. However, since reference [12] requires more time to manage and select supervisor group members, after the number of modification requests exceeds 28, our scheme begins to outperform that in [12] as the number of requests continues to rise.

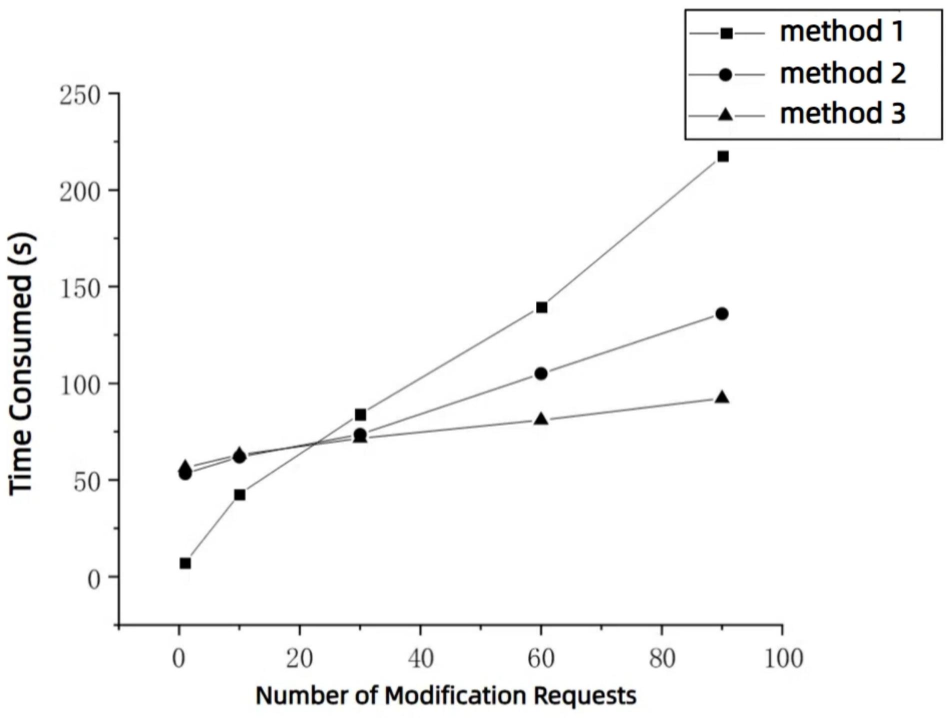


Figure 5. The modification efficiency when handling multiple modification requests simultaneously. (Method 1 references the method in [8], Method 2 references the method in [12], and Method 3 is the method used in this paper).

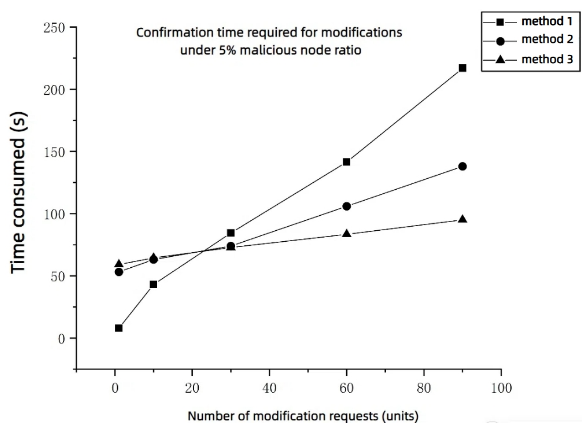
3. Confirmation Time Required for Modification under Different Proportions of Malicious Nodes

This experiment simulates the average modification time required for each scheme to process multiple requests simultaneously as the number of modification requests increases, under different proportions of malicious nodes. Our proposed scheme is compared with the schemes in references [8,12]. The experimental results are shown in Figure 6. The experiment tested scenarios where the proportion of malicious nodes accounted for 5%, 15%, 30%, and 40% of the total nodes. As the number of modification requests increased, the time taken by the three schemes to complete modification requests and confirm them was measured.

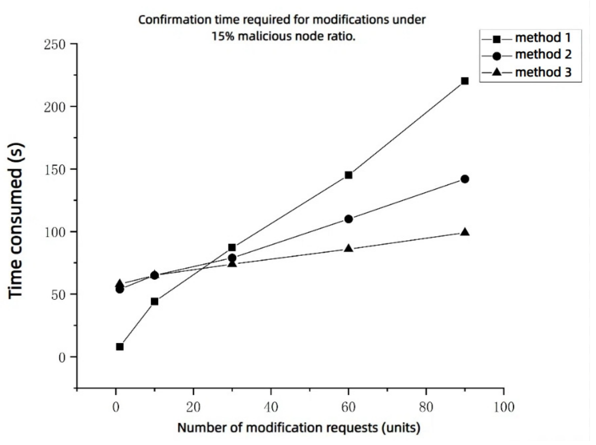
As shown in Figure 6, as the proportion of malicious nodes increases, the scheme in [8] is most affected, with the lowest modification efficiency. The scheme in [12] is affected to a lesser extent, while our proposed scheme is least affected and exhibits the best modification efficiency.

4. Reputation Value Changes of Different Nodes After Multiple Modifications

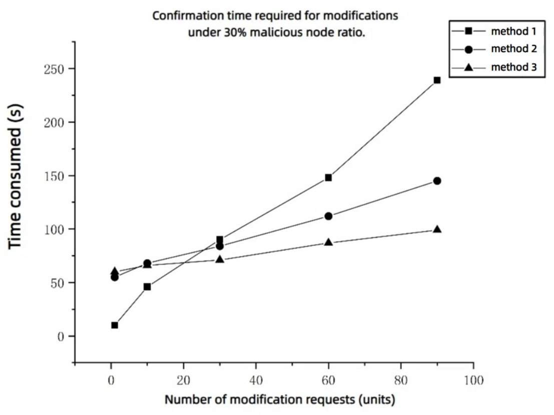
This experiment simulates the reputation value changes of a trustworthy node, a malicious node, and a passive node after participating in multiple modification processes within the proposed scheme. The experiment is set with a one-day interval between each modification, and the initial reputation values of the three nodes are all 60. The transaction cycle constant $T = 10$ and the growth rate $v = 1$ in Equation (3) are applied.



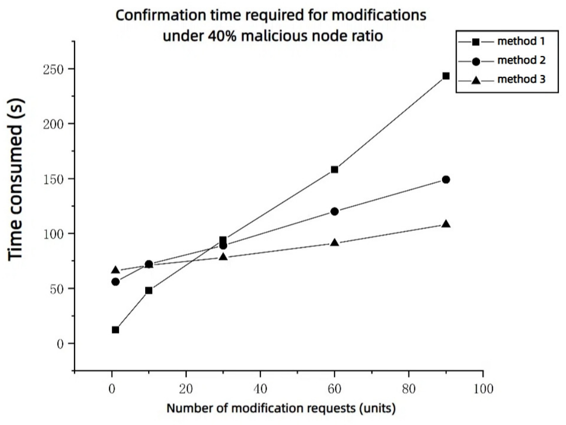
(**a**)



(**b**)



(**c**)



(**d**)

Figure 6. The confirmation time required for modifications under different proportions of malicious nodes. (**a**) Confirmation time required for modifications under a 5% malicious node ratio. (**b**) Confirmation time required for modifications under a 15% malicious node ratio. (**c**) Confirmation time required for modifications under a 30% malicious node ratio. (**d**) Confirmation time required for modifications under a 40% malicious node ratio. (Method 1 references the method in [8], Method 2 references the method in [12], and Method 3 is the method used in this paper).

As shown in Figure 7, the trustworthy node performs well in each modification process, and its reputation value gradually increases and stabilizes as the number of modifications increases. The malicious node, however, engages in malicious behavior during the first modification process, causing its reputation value to drop to 47, below the threshold $R_{\min} = 55$ for participation in future modification processes. As a result, it is prohibited from participating in subsequent modification processes.

The passive node shows inactivity during the first modification process (i.e., it is selected as a member of the gatekeeper management group N_{tk} but does not vote), leading to its reputation value dropping to 51. After applying Equation (3) to restrict its participation within a certain cycle, its reputation value recovers to 55.995, allowing it to continue participating in modification processes. However, during the seventh modification process, the passive node again fails to take action, and its reputation value drops to 46.995, below the threshold $R_{\min} = 55$, resulting in its exclusion from further modification processes.

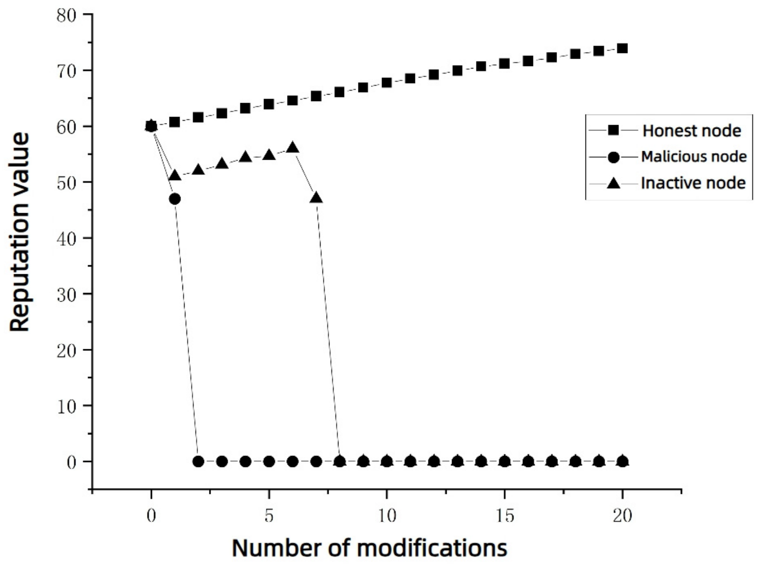


Figure 7. The change in reputation values of different nodes after multiple modifications.

In summary, both malicious and passive nodes are removed from the modification process after engaging in malicious behavior or inactivity, ensuring they will not pose a threat or impact the subsequent modification processes. This guarantees the safety and efficiency of the modifications. Meanwhile, the trustworthy node's reputation value shows a nonlinear upward trend and gradually stabilizes in the later stages, preventing the accumulation of excessive reputation values, which could lead to an overly centralized system. This ensures the decentralization characteristics of the entire blockchain system.

6. Conclusions

This paper proposes a modifiable blockchain solution based on the RE-TNG node selection method, aimed at addressing the issues of dishonest and inactive nodes affecting modification efficiency, modification rights, and the excessive centralization of trapdoor management in existing modifiable blockchain solutions. Additionally, it tackles the problem of reputation value accumulation in reputation-based node selection for modification. By combining the node reputation reward and penalty model with secondary grouping, the RE-TNG node selection method is designed. The Fibonacci sequence grouping rule is used to filter nodes participating in the modification process, effectively solving the issue of reputation value accumulation and ensuring long-term decentralization of the system. The second grouping selects high-reputation nodes, ensuring the integrity of nodes assigned modification rights or trapdoor management roles. This method effectively reduces the impact of dishonest and inactive nodes on modification efficiency. The multi-party collaboration for trapdoor key generation decentralizes trapdoor management, enhancing its security. For selecting the final modifier, the method uses a verifiable random function to randomly select and then employs zero-knowledge proofs to ensure accountability. Additionally, the proposed modifiable blockchain solution based on the RE-TNG node selection method has been analyzed and experimentally evaluated. The analysis and experimental results indicate that, compared to existing solutions, the proposed method offers a higher modification efficiency, security, and decentralization, while ensuring the editability and security attributes of the blockchain.

Although the RE-TNG grouping rule effectively selects trusted nodes, further achieving decentralization and avoiding the large communication overhead of involving all nodes in the selection, the initial grouping process still incurs considerable computational overhead. Therefore, future work will focus on optimizing computational algorithms, reducing the number of nodes initially participating in grouping by random selection, and reducing the communication consumption associated with reputation evaluation and two-stage grouping calculations during each modification. Regarding random selection algorithms, the plan is to use weighted random sampling or hash-based random selection, which will require further research and experimental verification.

Author Contributions: Conceptualization, S.Z.; methodology, C.L. and B.D.; software, R.C.; validation, R.C.; investigation, C.L. and B.D.; writing—original draft preparation, R.C.; writing—review and editing, R.C. and C.L.; supervision, C.L.; project administration, C.L. and S.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: The original contributions presented in the study are included in the article; further inquiries can be directed to the corresponding author.

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

References

1. Yuan, Y.; Wang, F.Y. Blockchain: The state of the art and future trends. *Acta Autom. Sin.* **2016**, *42*, 481–494.
2. Liu, S.Y.; Lei, M.Y.X.; Wang, L. A review of key technologies and existing problems in blockchain research. *Comput. Eng. Appl.* **2022**, *58*, 66–82.
3. Cai, X.Q.; Deng, Y.; Zhang, L. The principles and core technologies of blockchain. *Chin. J. Comput.* **2021**, *44*, 84–131.
4. He, S.; Huang, X.N.; Liu, Q.B.; Yang, Y.J. Research on improvement of DPoS blockchain consensus mechanism. *Appl. Res. Comput.* **2021**, *38*, 3551–3557.
5. Yuan, Y.; Wang, F.Y. Editable blockchain: Models, techniques and methods. *Acta Autom. Sin.* **2020**, *46*, 831–846.
6. Krawczyk, H.; Rabin, T. Chameleon Hashing and Signatures. Cryptology ePrint Archive. 1998. Available online: http://eprint.iacr.org/1998/010 (accessed on 5 October 2023).
7. Ateniese, G.; Magri, B.; Venturi, D.; Andrade, E. Redactable blockchain-or-rewriting history in bitcoin and friends. In Proceedings of the 2017 IEEE European Symposium on Security and Privacy, Paris, France, 26–28 April 2017; pp. 111–126.
8. Li, P.L.; Xu, H.X.; Ma, T.J.; Mu, Y. Research on fault-correcting blockchain technology. *J. Cryptologic Res.* **2018**, *5*, 501–509.
9. Ashritha, K.; Sindhu, M.; Lakhmy, K.V. Redactable blockchain using enhanced chameleon hash function. In Proceedings of the International Conference on Advanced Computing and Communication Systems, Coimbatore, India, 15–16 March 2019.
10. Lyu, W.L.; Wei, S.J.; Yu, M.H.; Li, S. Research on verifiable blockchain ledger redaction method for trusted consortium. *Chin. J. Comput.* **2021**, *44*, 2016–2032.
11. Gao, W.; Chen, L.Q.; Tang, C.M.; Zhang, G.; Li, F. One-time chameleon hash function and its application in redactable blockchain. *J. Comput. Res. Dev.* **2021**, *58*, 2310–2318.
12. Gu, K.; Zhang, S.H.; Li, C. Blockchain ledger revision scheme based on supervisor group. *Appl. Res. Comput.* **2023**, *40*, 2266–2273.
13. Chun, H.W.; Li, H.K.; Yu, S.D. Quantum resistant key-exposure free chameleon hash and applications in redactable blockchain. *Inf. Sci.* **2020**, *548*, 438–449.
14. Khalili, M.; Dakhilalian, M.; Susilo, W. Efficient chameleon hash functions in the enhanced collision resistant model. *Inf. Sci.* **2020**, *510*, 155–164. [CrossRef]
15. Shamir, A. How to share a secret. *Commun. ACM* **1979**, *22*, 612–613. [CrossRef]
16. Chor, B.; Goldwasser, S.; Micali, S.; Awerbuch, B. Verifiable secret sharing and achieving simultaneity in the presence of faults. In Proceedings of the 26th Annual Symposium on Foundations of Computer Science, Portland, OR, USA, 21–23 October 1985.
17. Chen, S.M.; Wang, B.; Chen, Y.Q. Improved PBFT consensus algorithm based on node grouping reputation model. *Appl. Res. Comput.* **2023**, *40*, 2916–2921.
18. Chen, R.Y.; Wang, L.W.; Zhu, R.G. PBFT consensus algorithm based on reputation value voting and random number election. *Comput. Eng.* **2022**, *48*, 42–49.
19. Sun, M.; Jiao, S.X.; Wang, C.Y. Credit-based committee consensus mechanism. *J. Comput. Appl.* **2024**, *1*, 1–10. Available online: http://kns.cnki.net/kcms/detail/51.1307.TP.20240409.1659.002.html (accessed on 14 May 2024).

20. Dai, B.R.; Jiang, S.M.; Li, D.W.; Li, C. Evaluation model of cross-chain notary mechanism based on improved PageRank algorithm. *Comput. Eng.* **2021**, *47*, 26–31.
21. Zhao, L.; Li, B.; Zhou, Q.L.; Chen, X. Improvement and optimization of consensus algorithm based on PBFT. In Proceedings of the 4th International Conference on Communications, Information System and Computer Engineering, Piscataway, NJ, USA, 27–29 May 2022; pp. 350–356.
22. Dodis, Y.; Yampolskiy, A. A verifiable random function with short proofs and keys. In *International Workshop on Public Key Cryptography*; Springer: Berlin/Heidelberg, Germany, 2005.
23. Zhao, X.; Zhang, Z.; Li, Y. An editable and accountable blockchain scheme. *J. Cyber Secur.* **2022**, *7*, 19–28. [CrossRef]
24. Lai, M.X.; Du, R.Y.; Chen, J.; He, K. A Decentralized and Traceable Editable Blockchain Scheme. *J. Wuhan Univ. Sci. Ed.* **2024**, *70*, 413–420. [CrossRef]