mathematics

Article

Enhancing Global Blockchain Privacy via a Digital Mutual Trust Mechanism

Sheng Peng [1,2], Linkai Zhu [3,*], Shanwen Hu [4], Zhiming Cai [5] and Wenjian Liu [4]

1 Academy of Management, Guangdong University of Science and Technology, Dongguan 523083, China
2 Zhuhai Yingying Technology Co., Ltd., Zhuhai 519080, China
3 Information Technology School, Hebei University of Economics and Business, Shijiazhuang 050061, China
4 Faculty of Data Science, City University of Macau, Macau 999078, China
5 Faculty of Digital Science and Technology, Macau Millennium College, Macau, China; zmcai@mmc.edu.mo
* Correspondence: linkai@hueb.edu.cn

Abstract: Blockchain technology, initially developed as a decentralized and transparent mechanism for recording transactions, faces significant privacy challenges due to its inherent transparency, exposing sensitive transaction data to all network participants. This study proposes a blockchain privacy protection algorithm that employs a digital mutual trust mechanism integrated with advanced cryptographic techniques to enhance privacy and security in blockchain transactions. The contribution includes the development of a new dynamic Byzantine consensus algorithm within the Practical Byzantine Fault Tolerance framework, incorporating an authorization mechanism from the reputation model and a proof consensus algorithm for robust digital mutual trust. Additionally, the refinement of homomorphic cryptography using the approximate greatest common divisor technique optimizes the encryption process to support complex operations securely. The integration of a smart contract system facilitates automatic and private transaction execution across the blockchain network. Experimental evidence demonstrates the superior performance of the algorithm in handling privacy requests and transaction receipts with reduced delays and increased accuracy, marking a significant improvement over existing methods.

Keywords: digital mutual trust mechanism; blockchain; global privacy protection algorithm; dynamic Byzantine consensus algorithm; fully homomorphic encryption

MSC: 68M25

check for updates

Citation: Peng, S.; Zhu, L.; Hu, S.; Cai, Z.; Liu, W. Enhancing Global Blockchain Privacy via a Digital Mutual Trust Mechanism. *Mathematics* **2024**, *12*, 1481. https://doi.org/10.3390/math12101481

Received: 17 March 2024
Revised: 4 May 2024
Accepted: 8 May 2024
Published: 10 May 2024

1. Introduction

Bitcoin, as a pioneering peer-to-peer electronic currency system, employs a public ledger known as the blockchain to record all transactions. This decentralized ledger ensures that every full network node retains a copy of all Bitcoin transactions, fundamentally distinguishing it from traditional centralized systems [1]. While this decentralized architecture enhances system robustness by preventing single points of failure and centralized data breaches [2], it also introduces significant privacy challenges. The inherent requirement for transaction transparency among all participating nodes exposes sensitive transaction and account balance information, compromising user confidentiality [3].

In recent studies, various researchers have outlined advanced privacy-enhancing techniques within blockchain frameworks to tackle transaction privacy issues effectively. For example, Wang and Liao [4] introduced a blockchain privacy protection algorithm that utilizes Pedersen commitment and zero-knowledge proofs to obscure transaction details while preserving their verifiability. In another significant contribution, An et al. [5] proposed the TCNS framework. This framework aims to enhance privacy in crowdsensing by employing a dual verification process through twice consensuses of blockchain, ensuring robust user attribute protection. Furthermore, Garcia et al. [6] developed a blockchain-based data governance framework tailored for electronic prescriptions in healthcare. Their work leverages blockchain's provenance capabilities to safeguard privacy, showcasing the technology's adaptability in sensitive sectors such as healthcare. These diverse approaches demonstrate blockchain's potential in enhancing data privacy across various fields. Steffen et al. [7] present ZeeStar, a programming language and compiler framework that facilitates the implementation of private smart contracts on Ethereum. This framework utilizes homomorphic encryption and non-interactive zero-knowledge proofs to offer a robust privacy mechanism, allowing developers to create contracts with enhanced privacy constraints efficiently. Chen et al. [8] introduce a Decentralized Privacy-preserving Deep Learning (DPDL) model for Vehicular Ad Hoc Networks (VANETs), which integrates blockchain and fully homomorphic encryption to safeguard data privacy. This model decentralizes computational tasks to edge nodes, thereby minimizing network overhead and enhancing data security during both transmission and analysis. Ali et al. [9] propose a hybrid deep learning model for the Industrial Internet of Medical Things (IIoMT), which uses homomorphic encryption and a consortium blockchain to secure Electronic Medical Records (EMRs). This approach provides a robust framework for performing statistical and machine learning operations on encrypted data, addressing privacy and security challenges prevalent in healthcare applications. Wu et al. [10] developed BPF-Payment, a blockchain-based fair payment system for cloud computing that utilizes the Paillier homomorphic encryption scheme. This system ensures transaction fairness and data privacy between users and service providers, exemplifying the integration of blockchain as a trustworthy intermediary in cloud service transactions. Jia et al. [11] proposed a Blockchain-Enabled Federated Learning Data Protection Aggregation Scheme that employs differential privacy and homomorphic encryption. This scheme is designed for the Industrial Internet of Things, providing secure data aggregation while mitigating risks associated with model extraction and reverse engineering attacks in federated learning environments. Tang et al. [12] address computational efficiency in additively homomorphic encryption systems. They introduce an efficient algorithm for solving the Elliptic Curve Discrete Logarithm Problem, thereby enhancing the decryption process, which is pivotal for practical applications in blockchain and federated learning that require rapid and secure data decryption.

These scholarly contributions demonstrate significant strides in the realm of blockchain privacy and security. Each addresses specific limitations of preceding systems and paves the way for the practical application of secure, scalable blockchain technologies across various industries. Our work leverages these foundational studies to introduce novel methodologies that tackle persistent challenges in blockchain privacy, especially focusing on previously unaddressed issues related to the visibility of transaction elements within blockchain systems.

However, despite these innovations, none have comprehensively mitigated the visibility issues associated with inputs, outputs, account balances, and transaction details in blockchain systems.

This paper introduces a novel global privacy protection algorithm that centers on a digital mutual trust mechanism, aiming to reconcile privacy with the integrity and efficiency demands of blockchain technology. Our primary contributions are as follows:

1. We integrate the reputation model and proof of stake within a Practical Byzantine Fault Tolerance (PBFT) framework, and we propose a dynamic Byzantine consensus algorithm that enhances the scalability and efficiency of node participation in consensus, addressing one of the primary limitations of traditional PBFT implementations.
2. We refine homomorphic encryption techniques using an approximate greatest common divisor method. This advancement enables the encryption of transaction details and account balances without compromising the ability to perform computations on encrypted data, thereby maintaining functionality alongside privacy.

3. By incorporating smart contracts into our privacy-focused blockchain framework, we automate the execution of transactions and the management of privacy settings, significantly reducing potential delays in privacy request handling and transaction processing.

2. Digital Mutual Trust Mechanism

Alliance chains are commonly referred to as industry memory blockchains or local area chains. The characteristics of an alliance chain are as follows: Firstly, the alliance chain achieves local decentralization. The organizations or enterprises participating in the alliance chain are decentralized, but appear as a whole externally. The second is the hierarchical management mechanism for data permissions. An alliance chain is formed by several enterprises or groups, which is only open to specific organizations and groups and only exists at organizational nodes. The trust between organizational nodes is high, and their data read and write permissions are managed at the internal level of the alliance. The third is that data can be modified, but their modification requires consensus from nodes within the alliance. The fourth is that the scope of participation in the alliance chain is not entirely shared by the parties of common interest. The alliance chain is currently mainly used in banking, product traceability, supply chain finance, and so on. For information asymmetry and malicious node behavior, the proposed method introduces a reputation model and authorization mechanism in a proven consensus algorithm [13,14] in a practical byzantine fault tolerance (PBFT) consensus algorithm, integrates PBFT and delegated proof of stake (DPoS) ideas [15], proposes a new dynamic Byzantine consensus algorithm, and then establishes digital mutual trust mechanism.

2.1. Consensus Algorithm Based on Reputation Authorization Certificate

DPoS equity authorization proves that consensus algorithms can effectively solve the problem that PBFT nodes cannot be dynamically changed, and consensus efficiency grows in a square order with the number of nodes. However, a DPoS consensus algorithm has the problem that nodes have low enthusiasm for voting and malicious nodes cannot quickly handle it. Therefore, the proposed method proposes a reputation proof of stake consensus algorithm, introduces a reputation model to replace equity, and realizes the rapid elimination of malicious nodes by setting reputation values and node states [16].

2.1.1. Node Reputation Model

Reputation value is a way to express the credibility of nodes. At the same time, reputation value affects the discourse power of consensus nodes. With a higher reputation, the more say nodes will have, and the greater the probability of obtaining the right to participate in block generation. The process of node participation in representative node selection or block generation is collectively referred to as consensus. The node reputation value comes from the performance of the historical consensus participation process and reflects the current status of the node as much as possible, that is, the more recent the consensus performance is, the greater the impact on the current reputation value [17]. In order to prevent the node reputation value from being too high and causing centralization, and to reflect the fluctuation of the node reputation value as much as possible, set the maximum credit value max of this node and obtain the i credit value R_i of this node, as follows:

$$\begin{cases} R_i = \min \frac{R_0 + \sum_{k=1}^{j} \left[TS_i \cdot r_i^k \cdot \varphi(k) \right]}{\sum_{k=1}^{j} \varphi(k), \max}, j = 2, 3, \cdots \\ \varphi(k) = \lambda^{i-k} \end{cases} \tag{1}$$

Among them, R_0 represents initial credit, TS_i represents the i transaction resource of a node, j represents i participants in a node, $\varphi(k)$ represents the decay function, λ represents any number in (0, 1), and r_i^k represents the change value of the reputation value when the node participates in the k consensus. If the node consensus process correctly generates blocks or selects the correct node as the representative node, $r_i^k > 0$ will be used. Reputation value decreases with the order of participating consensus, that is, the weight of early consensus on current reputation value is reduced and the weight of recent consensus is increased. The trustworthiness of R determines the trustworthiness of a consistent node. The proposed method divides the status ST of the consensus node into the following four categories according to the threshold value:

$$\begin{cases} Excellent : R_i > \alpha, ST = 3 \\ Good : \alpha \leq R_i < \beta, ST = 2 \\ Normal : \beta \leq R_i < \gamma, ST = 1 \\ Error : R_i < \gamma, ST = 0 \end{cases} \tag{2}$$

The parameters α, β and γ meet $\max \geq \alpha > \beta > \gamma \geq 0$. To avoid nodes with low reputation value participating in consensus, their rights are classified according to their credit status. For each new node added to the network, the initial state is the lowest value of Normal, namely $R_0 = \beta$. Among the typical nodes, typical nodes can be selected only when they are Normal or above that level. Only nodes with a Good and above status can be selected as representative nodes. Consensus nodes can be divided into two categories: representative and alternative. Only nodes with an Excellent status can be selected as representative nodes. For nodes with an Error status, the system will clear them.

2.1.2. RPoS Consensus Algorithm

In RPoS, select the representative node according to the credit degree of the node. Nodes with higher credit are easier to select. Firstly, the N node with the largest voting credit is selected as the representative node. If a node is found to generate malicious blocks or act with low enthusiasm, the node will lose its reputation and representative node identity. The RPoS consensus process in a cycle includes three modules: select a certain number of representative nodes based on the credit value; remove malicious nodes; represent node scheduling and block generation. In RPoS, there are two main types of nodes: common node and consistent node. The consensus node is divided into a candidate node and a representative node, and the common node mainly executes transactions. The subnet environment composed of consensus nodes is called the consensus network, and the network environment composed of ordinary nodes is called the transaction subnet. The nodes in the transaction subnet can participate in the consensus network according to their wishes on the premise that the node status is Normal.

N represents the set of nodes in the network, k represents the total number of nodes, N_i represents the ith node, $i \in [1, k]$, NC represents the set of consensus nodes, k_1 represents all negotiation nodes, NC_i represents i, NT represents transaction nodes, k_2 represents the number of transaction nodes, and NT_i represents the i transaction node, so the following can be obtained:

$$\begin{cases} N = NC \cup NT \\ k = k_1 + k_2 \end{cases} \tag{3}$$

C_{ij} represents consensus node i voting on consensus node j; C_{ij} is defined as follows:

$$C_{ij} = \begin{cases} 1, \text{Node i votes for node j} \\ 0, \text{Node i did not vote for node j} \end{cases} \tag{4}$$

R_j represents the j reputation value of the node, and ST_i represents the i status of the node. The total reputation value SR_j of the node j and the number of votes m_j of the node j are as follows:

$$\begin{cases} SR_j = \sum_{i=1}^{k_1} \left(R_i \cdot C_{ij} \cdot ST_i \right) + R_j \\ m_j = \sum_{i=1}^{k_1} C_{ij} \end{cases} \tag{5}$$

Representative node selection requires selecting the top N of the total reputation ranking under the premise of $m_j \geq k_1/2 + 1$. When a node participates in the selection or consensus process of representative nodes, its reputation value will change accordingly.

The change in reputation value r_i^k is related to the current status, enthusiasm, and malicious degree of the node, as shown below:

$$r_i^k = ST_i \cdot e^{\Delta t} \cdot \eta(k) \tag{6}$$

In the formula, Δt represents the interval between the last two times node i participated in the representative node selection or consensus process. When node i is positive, Δt is negative, and vice versa. $\eta(k)$ is the fluctuation amplitude, which is related to the number of consecutive positive performances of the node. When the node is positive, $\eta(k)$ is positive, and vice versa. The fluctuation amplitude in the consensus process is greater than the absolute value of the fluctuation amplitude in the process of selecting the representative node.

2.2. Dynamic Byzantine Consensus Algorithm Based on PBFT-RPoS

Although PBFT has Byzantine Fault Tolerance, high efficiency, and low power consumption [18–20], it also has shortcomings such as malicious nodes that cannot be eliminated in time and C/S response mode; since the number of nodes in the network cannot be dynamically sensed, the efficiency of consistency will decline when the number of nodes increases. To solve these problems, the PBFT consistency protocol and master node scheduling are optimized based on RPoS, and a dynamic Byzantine consensus algorithm (DPBFT) is proposed.

2.2.1. PBFT Consensus Algorithm

PBFT is a state machine replication algorithm, which can ensure system security and activity when the number of failed nodes is less than or equal to $f = (n-1)/3$, where n represents the total number of nodes in the network. The PBFT algorithm divides the network into main nodes and other nodes. The basic process is as follows:

1. The client sends the business request to the master node.
2. The main node transfers the requirements to another node, while the other nodes handle them uniformly.
3. After processing the request, other nodes return a message to the client.
4. When the client receives the same information from $f+1$ nodes, the request is executed.

The consensus of PBFT consists of three steps: preliminary preparation, preparation, and confirmation. The algorithm consensus process is implemented in the view. There is a primary node in the node, and other nodes are backup nodes. Figure 1a shows the consensus operation process of the PBFT consensus algorithm in a distributed network with four nodes and one malicious node. The specific steps are as follows:

1. Request stage: a requester sends a request to the master node, and the request information is m;
2. Pre-preparation stage: the host sorts the requests received by the client, and then broadcasts the pre-preparation message $\langle\langle PRE-PREPARE, View, n, D(m)\rangle, m\rangle$ to each backup node, where m is the request message sent by the client, and $D(m)$ is the summary of the request message m;
3. Preparation stage: When the copy node receives the pre-prepared information, it checks the validity of the message according to $D(m)$, $View$, and n. If it is confirmed to be true, it sends the preparation message $\langle PREPARE, View, n, D(m), i\rangle$ to other nodes; i is the replica node number and receives preparatory information from other replica nodes. If the number of received preparation messages reaches $2f+1$, the node enters the preparation state;
4. Confirmation phase: the replica node broadcasts the confirmation message $\langle COMMIT, View, n, D(m), i\rangle$, notifies other nodes that they are ready, and receives the confirmation message from other replica nodes. When the number of confirmation messages received reaches $2f+1$, the request is completed;

5. Feedback stage: all backup nodes send feedback to the client, and the client will receive the same $f+1$ results from different nodes as the final result.

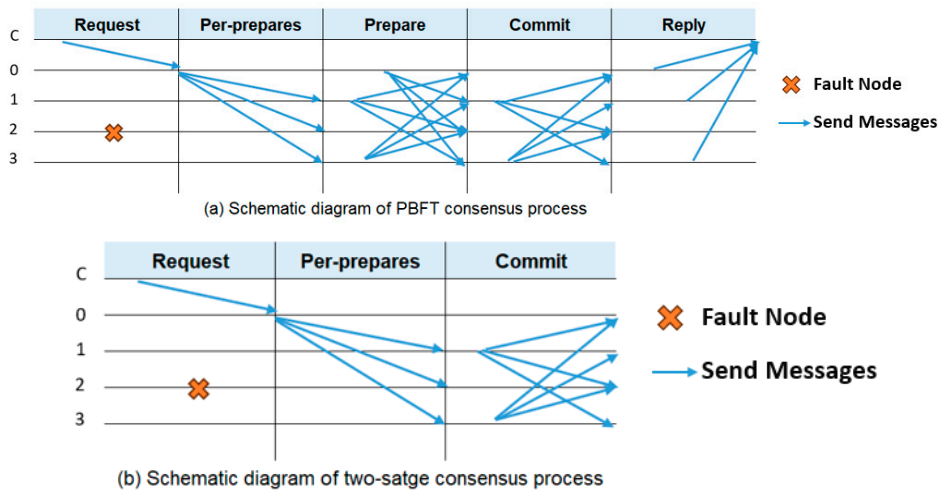

Figure 1. PBFT consensus and two-phase consensus process diagram.

2.2.2. Improvement of Consistency Protocol

The PBFT algorithm uses a three-segment protocol to ensure that each replica node has the same execution order. The three-segment protocol includes one single-node broadcast and two all-node network broadcasts [21]. The number of consensus messages transmitted is $Z_1 = 2n^2 - n$, and n is the total number of nodes in the entire network.

The complete consistency protocol has two communication processes with $O(n^2)$ complexity. Moreover, PBFT is a state machine replica replication algorithm. The master node needs to collect and sort the requests and then distribute them to the replica nodes. In essence, it is a C/S response mode, which is not suitable for P2P environments [22].

The consensus node is elected by reputation authorization voting, and RPoS can eliminate malicious nodes according to node reputation; therefore, the consensus nodes can be regarded as trusted.

Therefore, a consistency protocol with less time complexity is proposed, as shown in Figure 1b. The specific steps are as follows:

1. Request phase: the requester broadcasts and sends request messages. All nodes sort the request messages in chronological order and verify the validity of the messages;
2. Preparation stage: each replica node compares the information of the master node with the information collected by itself, and broadcasts the verification information if it passes the verification;
3. Confirmation phase: when each node receives the correct verification message from no less than $2f$ nodes, the request is completed.

When implementing the simplified protocol in a network with n nodes, the number of times for consensus message propagation is $Z_2 = n^2 + n$, and the number of times for consensus message propagation reduction is $\Delta Z = Z_1 - Z_2 = n^2 - 2n$. Compared with that before simplification, the simplified consistency protocol has changed from three segments to two segments, reducing $O(n^2)$ in number. In the case of a large number of consensus nodes, the bandwidth requirements of the network can be greatly reduced, and the consistency between nodes can be greatly improved.

2.2.3. Dynamic Byzantine Consensus Algorithm

The dynamic Byzantine consensus algorithm combines the consensus algorithm of reputation authorization proof on the basis of PBFT to establish the node reputation model. The consensus process includes the following three aspects: consensus node selection, master node scheduling, and consensus protocol. On this basis, the earliest N node is selected for negotiation through authorization voting on the node credit value. Through the shuffling algorithm and node status, the master node can rotate within a week, and the consistency protocol is used to achieve data consistency in the distributed network.

The shuffling algorithm is used to disrupt the order of the selected N consensus nodes and then schedule the master node [23]. DPBFT adopts the scheduling mechanism of the master node to ensure the security and dynamic performance of the distributed system. PBFT selects the next master node by rotation to predict in advance that the next master node may bring risks. Therefore, the proposed method proposes to select the node with the most continuous positive behaviors as the master node. The more times a node has continuous positive behaviors, the more likely it is to select it as the primary node. The possibility of selecting this node as the main node P_i is as follows:

$$P_i = \frac{m_i \cdot ST_i}{\sum_{j=1}^{k}(m_i \cdot ST_i)} \tag{7}$$

where m_i represents the number of consecutive positive behaviors. In multiple nodes, when the probability value and the probability of $ST = 3$ are the maximum, the random number is used to determine the master node RN.

$$RN = (n \oplus Timestamp)\text{mod}N \tag{8}$$

where n represents the number of nodes in the current network, and N represents the number of scheduling times. After the master node schedules $N/2$ times, the existing consensus node is cleared, and the new consensus node is authorized based on credit.

Node consensus is mainly the process of building the data consistency of distributed network nodes. In the request phase, all consensus nodes sort the transaction requests in chronological order, simplifying the original three-segment PBFT protocol into a two-segment protocol and reducing the time and bandwidth consumption caused by the consensus process [24].

Through the above digital mutual trust mechanism, we can quickly generate blocks and build blockchains, using the secure mutual trust blockchain as the basis for subsequent global privacy protection. In addition, the password algorithm and privacy are protected in the blockchain.

3. Methodology

Through a detailed examination, we aim to elucidate the intricate steps and innovative approaches proposed by the authors to address the pressing challenges surrounding privacy in blockchain systems. From an exploration of existing limitations to the introduction of novel techniques such as the digital mutual trust mechanism and dynamic Byzantine consensus algorithm, each facet of the proposed methodology will be scrutinized.

The method begins by discussing the intrinsic transparency of blockchain as implemented in systems. This transparency, while ensuring data integrity and preventing fraud, simultaneously compromises user privacy by making all transactions visible on the public ledger. The need for enhanced privacy mechanisms is clear, especially in compliance with financial confidentiality norms and regulatory standards. It identifies the limitations of current privacy-enhancing techniques in blockchain systems. These include the inadequate concealment of transaction details, account balances, and participant identities, which do not satisfy the strict privacy requirements of many potential blockchain applications.

To address these challenges, this paper proposes a novel digital mutual trust mechanism. This mechanism integrates an authorization system derived from a reputation model, where nodes in the blockchain are evaluated and authorized based on their historical behavior and contributions to the network integrity. The reputation model assigns a quantitative value to each node's trustworthiness, influencing its role and weight in the consensus process. Nodes with higher reputation scores have a greater influence in decision-making processes, such as transaction validation and block creation. The core of the proposed method is the development of a new dynamic Byzantine consensus algorithm that incorporates the reputation-based authorization within the Practical Byzantine Fault Tolerance (PBFT) framework. This enhancement allows for more dynamic and efficient handling of node failures and malicious activities, thus improving overall system resilience.

To ensure that the privacy of data is maintained even during processing, this paper refines homomorphic encryption techniques using the approximate greatest common divisor (GCD) method. This advancement allows the blockchain to perform encrypted calculations without decrypting, maintaining confidentiality while enabling useful computations like balance checks and transaction validations. The integration of smart contracts is another pivotal aspect of the proposed architecture. These contracts automatically execute transactions according to predefined rules and conditions encoded in the blockchain, facilitating seamless and private transfers of value without human intervention. The practical implementation of these concepts results in a privacy-focused blockchain framework. This framework encrypts all transaction details and only reveals information necessary for the validation process, thus maintaining a high level of confidentiality.

The effectiveness of the proposed system is validated through rigorous experiments. These experiments demonstrate the capability of the privacy protection algorithm to handle inter- and intra-chain transactions efficiently while maintaining the privacy of the transaction details against various threat models. Specific attention is given to privacy protection across different blockchain partitions. The system ensures that data integrity and privacy are maintained even when transactions cross the boundaries of these partitions, which is critical for applications involving multiple blockchain networks. Within a single blockchain or partition, the system prevents unauthorized access to transaction data, ensuring that only participants who are directly involved in a transaction have access to their details. This granular level of control is crucial for maintaining confidentiality in sensitive business environments.

4. Blockchain Global Privacy Protection Algorithm

4.1. Homomorphic Encryption Technology

In homomorphic cryptography, the homomorphic cryptographic algorithm is represented as $HE = (keyGen, Enc, Dec, Eval)$ [23], where $HE.keyGen(1^{\lambda})$ represents a key generation algorithm, inputs security indicators λ, and outputs the public key pk, key sk, and bootstrap public key evk, namely $(pk, sk, evk) \leftarrow keyGen(1^{\lambda})$; $HE.Enc(pk, m)$ represents the encryption algorithm. Input pk and plaintext m, and output ciphertext c, i.e., $c \leftarrow Enc(pk, m)$; $HE.Dec(sk, c)$ represents the decryption algorithm. sk and c are input and output, that is, m^*. $m^* \leftarrow Dec(sk, c)$; $HE.Eval(evk, f, c)$ represents the evaluation algorithm. Enter pk, function f and l ciphertext $c = (c_0, c_1, \cdots, c_{l-1})$. i ciphertext c_i corresponds to i plaintext m_i, and the output fresh ciphertext c^*, namely $c^* \leftarrow Eval(evk, f, c)$, $Dec(sk, c^*) = f(m), f \in \updownarrow$, $\updownarrow$, represents the operation circuit in the plaintext space. f meeting the above conditions belongs to the permitted function, that is, the operation function f can correctly perform homomorphic operations. If any function f can meet the conditions, the scheme is called an all-homomorphic encryption scheme [24].

In the traditional integer homomorphic encryption algorithm DGHV, the symmetric encryption scheme of the partially homomorphic encryption scheme is $c \leftarrow m + 2r + pq$, where r is a small integer randomly generated in the encryption process, p is the key, and q is a large integer generated in the key generation phase, $m \in \{0, 1\}$. The decryption algorithm is $(c\mathrm{mod}p)\mathrm{mod}2 \leftarrow Lsb(c) \times or\lceil c/p \rfloor$, where Lsb is the least significant bit [25] and $\lceil \cdot \rfloor$ is rounded. Due to the interference of r, this encryption scheme cannot correctly identify the value of plaintext m, so $m + 2r$ is called noise. When the noise value is greater than $p/2$, it cannot be decrypted correctly. Therefore, $m + 2r < p/2$ must be guaranteed before decryption.

The parameter a is introduced to define $a\mathrm{mod}p = a - \lceil a/p \rfloor \times p$. In the decryption algorithm of the above scheme, the range of $c\mathrm{mod}p$ is $(-p/2, p/2]$. When the value of $c\mathrm{mod}p$ is greater than $p/2$, it cannot be decrypted correctly. In the process of homomorphic operation, the noise will increase with the operation, resulting in the ciphertext obtained after the operation not being decrypted correctly. Therefore, we need to encrypt again to update the ciphertext [26].

If the function f in the $HE.Eval(\cdot)$ algorithm can be executed correctly, $HE.Eval(\cdot)$ belongs to permitted function, namely $HE.Eval(evk, f, c)$. The result decrypted after the f operation is exactly the result of the f corresponding plaintext operation. If the decryption algorithm belongs to the permitted function, it can perform decryption in the $Eval$ algorithm, that is, $HE.Eval(pk_2, Dec, sk'_1, c')$. The specific process of re-encryption is as follows: The sk'_1 is obtained by encrypting the private key sk_1 of the first layer with the public key pk_2 of the second layer. Use pk_2 to encrypt ciphertext c to obtain c' and decrypt sk'_1 and c'; the result c^* is equivalent to that of plaintext m encrypted by pk_2, and c^* is a fresh ciphertext, which can definitely be decrypted correctly. The whole process is equivalent to refreshing ciphertext c to ciphertext c^*. However, the homomorphic encryption algorithm above can only encrypt one bit of plaintext at a time, so the proposed method improves the scheme and proposes a scheme that can encrypt n bits of plaintext at one time. First, in the key generation phase, a key p and large integer q are generated according to the security parameter λ. The encryption algorithm is $c \leftarrow m + 2^n r + pq$, where n is the number of bits encrypted at one time, and p and q are generated in the key generation phase. In the process of encrypting plaintext, the n value is determined according to the specific plaintext, and the n value is the same. In order to ensure the correctness of decryption, ensure $m + 2^n r < p/2$. The decryption algorithm is $m \leftarrow (c \bmod p) \bmod 2^n$.

However, in the above solutions, if pq is used as the public key, you can easily find the private key p. Therefore, the proposed method introduces the biggest convention problem to the above encryption algorithm [27], that is, adding some ciphertext $\{x_i; x_i = 2^n r_i + pq_i\}$ encrypted with plaintext as 0, using the set as the public key, and adding the sum of some subsets of the set randomly to the encryption algorithm during encryption to ensure the security of the scheme. Since the added ciphertext is 0, it has no effect on decryption.

The approximate maximum common divisor problem is still a mathematical problem that cannot be solved at present, and any attack on this scheme can be converted into a solution to the maximum approximate common divisor problem, so it is determined that this scheme is safe. Combined with the encryption scheme of shortening the size of the public key, the approximate GCD problem is introduced and the encryption algorithm is improved as follows:

$$c \leftarrow m + 2^n r + 2^n \sum_{1 \le i,j \le \sqrt{\tau}} b_{i,j} x_{i,0} x_{j,1} \tag{9}$$

In the formula, $p \in \left[2^{\eta-1}, 2^{\eta}\right]$ determines $b \in \{0,1\}$, $x_{i,b} = pq_{i,b} + r_{i,b}$, $i \in \left[1, \sqrt{\tau}\right]$, η and τ as the adjustment parameters in the key generation phase, the public key is $pk = \left(x_0, x_{1,0}, x_{1,1}, \cdots, x_{\sqrt{\tau},0}, x_{\sqrt{\tau},1}\right)$, and the number of public keys is $2\sqrt{\tau}$. In the key generation phase, a random vector $b = \left(b_{ij}\right)$, $b_{ij} \in \{0,1\}$ is generated. In this scheme, the public key size is reduced to $2\sqrt{\tau}$. The decryption algorithm is as follows:

$$(c \bmod p) \bmod 2^n \tag{10}$$

Therefore, a secure homomorphic encryption scheme is obtained and used in the subsequent blockchain global privacy protection.

4.2. Homomorphic Encryption Global Privacy Protection Algorithm

The scheme adopts a homomorphic encryption algorithm to encrypt transactions and accounts in the blockchain, and the account balance is stored in the blockchain as a password file. Using $E(v_1), E(v_2), \cdots, E(v_t)$ to represent t ciphertext with a balance of $v_1, v_2, \cdots, v_t$, miners can effectively calculate the ciphertext of $f(v_1, v_2, \cdots, v_t)$ [28]. The proposed method builds a global privacy protection blockchain based on smart contracts; the system can not only cover up the input, output, and transaction details of the traditional blockchain trading system but also enable the smart contract running on the blockchain to process the plaintext information, expand the blockchain application scenario, and truly achieve global privacy protection [29].

4.2.1. Privacy Blockchain

The private blockchain transaction information is encrypted into ciphertext using the homomorphic encryption algorithm, and the transaction ciphertext information is stored in the data block. Each transaction content includes the address of the transaction initiator and the address of the transaction receiver; the transaction initiator encrypts the account balance with its own public key, and the transaction amount needs to be encrypted with the public key of the transaction receiver, the non-interactive zero-knowledge certification π that proves the transaction is reasonable, the number of transfer currencies specified under the ciphertext state, and the transaction-related substrings. Among them, π includes the non-interactive zero-knowledge proof π_1 provided by the fully homomorphic encryption blockchain equivalent transaction protocol and the minimum necessary zero-knowledge proof π_2 provided by the fully homomorphic encryption blockchain transaction protocol.

The data block structure of the transaction data privacy protection blockchain is similar to Ethereum [30]; the block consists of three parts: block, uncle block, and transaction list. The block contains all the hash values of the previous block [31], the hash values of all the blocks in the block, the hash of all the transaction lists in this block, the hash of all the transaction receipt lists in this block, the difficulty level of this block, the block serial number, the timestamp created in this block, the random number, etc. Tertiary blocks are isolated blocks. Since the transaction data privacy protection blockchain may produce multiple legal blocks at the same time, in order to ensure security, competitive blocks are allowed to hang on the main chain, and isolated blocks can be up to 6 heights. The transaction list stores all transactions in this block. The private blockchain uses smart contracts to create transactions as follows:

1. Transaction originator A sends a request to establish a smart contract. When initiating a transaction request, the transaction ciphertext information and π are jointly published in the private blockchain network, and the private blockchain then creates a smart contract transaction request;
2. Node V on the network, i.e., user V, synchronizes the transaction and verifies whether the transaction is valid according to the π provided by transaction initiator A. User V places the verified transaction request into the transaction storage pool and forwards it to other nodes. Other nodes receiving the transaction request repeat the process of user V;
3. The miner packs this transaction and other transactions into the block and runs the called contract code on the local EVM until the end of the code operation;
4. User V sends the block containing the transaction request of transaction initiator A to the peer node and spreads it throughout the network;
5. The consensus node verifies the rationality of all transactions in the block after receiving the block. If the block passes the verification, the node will delete the request of original transaction initiator A to create a smart contract transaction in the memory pool, synchronize the private blockchain, and deploy the smart contract in their local private blockchain.

4.2.2. Smart Contracts on the Privacy Blockchain

In the process of a Bitcoin transaction, the transaction initiator must indicate in signed words that it is a legitimate user of UTXO and use the output script to specify the transaction receiver. At the start of each transaction, a signature for authentication needs to be generated using the private key, and for each transaction, the total input cannot be less than the total output. Ethereum transactions are similar to Bitcoin. Ethereum uses an account/balance model, and the status can be stored in the account in real time. This account model greatly facilitates smart contracts.

A smart contract is a piece of code stored in each node participating in the blockchain network, which can be regarded as an electronic version of a traditional contract. Suppose that in a payment system with an account architecture, transaction initiator A wants to transfer t coins to transaction receiver B, and transfer operations and some necessary checks can be deployed in the blockchain. Transaction initiator A can publish the transaction as follows: "Transfer t coins of A to user B, and σ is a signature of t". A smart contract is activated through transaction information to verify the correctness of a signature. If A has more than t coins, it will transfer money and publish transaction information on the blockchain. Otherwise, the transaction will be ignored.

In the above simple transaction, anyone can learn that t coins are transferred from user A to user B. Therefore, based on the traditional blockchain transaction model, the proposed method solves this problem by encrypting transaction information with homomorphic encryption. This paper proposes a $FHE.Enc(\cdot)$-based homomorphic password algorithm, which saves the balance of each user account as a password file to a blockchain account book. Transaction initiator A can publish the transaction as follows: "Transfer $FHE.Enc(t)$ coins of A to user B". The transaction information is a zero-knowledge proof π without interaction, which can prove that $FHE.Enc(t)$ is correct and the balance on the account is more than t. Other nodes on the blockchain verify the transaction information. If the confirmation is correct, a confirmed transaction will be published. On a blockchain, an intelligent contract will be automatically transferred. The smart contract updates the account balance password of transaction initiator A and the account balance of transaction initiator B according to the execution results. After a period of time, the protocol will be added to the blockchain together with the new blockchain. If not, the transaction will be ignored by the verification node.

4.2.3. Build a Blockchain Scheme to Protect Transaction Data Privacy

In the account mode, the transaction information in the blockchain is encrypted using a homomorphic encryption algorithm, and only the balance of the account is saved in the blockchain as a password file. When a transaction initiator passes a transaction message through a blockchain, it needs to provide a zero-knowledge proof π that proves the transaction is reasonable. Other verification nodes on the blockchain calculate according to the π verification. If the transaction information is confirmed, the smart contract will be executed correctly on the blockchain. The smart contract will update the password of the account according to the execution result; otherwise, it will be ignored. The specific implementation process of the global privacy protection blockchain is as follows:

1. Setup phase

The public parameters of the data privacy protection scheme are generated in the setting phase. In this stage, the security parameters of the homomorphic encryption scheme and the computing circuit depth that the homomorphic encryption can handle are used as inputs to generate the public parameter *params* of homomorphic encryption without bootstrap conversion.

2. User initialization phase

In the initialization phase, the user information is generated by the homomorphic encryption algorithm. To simplify the algorithm, users only consider transaction initiator A, transaction receiver B, and other third-party verification nodes V in the global privacy protection blockchain. The input in this stage is *params*; the user's public key, private key, and password information are then generated. You can use the public key of the consumer as the transaction collection address, and the private key can generate a digital signature σ.

Because the transaction is composed of the transaction originator and the transaction receiver, in this process, the data security mechanism must generate the public key and private key, respectively. When transaction initiator A initiates a transaction, user A's public key pk_A, private key sk_A, and account balance t_A's ciphertext C_{A,t_A} need to be generated at this stage, including $C_{A,t_A} = FHE.Enc(pk_A, t_A)$. Only the ciphertext C_{A,t_A}

of transaction initiator A is stored on the blockchain ledger. In the transfer transaction stage, it is necessary to generate a non-interactive zero-knowledge proof π_1 to prove that the transaction amount is equal and a non-interactive zero-knowledge proof π_2 to prove that the transaction account amount is sufficient. Therefore, in the initialization phase, A needs to encrypt t_A, t and t_{A-t} with its own public key to generate ciphertext C_{A,t_A}, $C_{A,t}$ and C_{A,t_A-t}, and A needs to encrypt transaction amount t with B's public key to generate ciphertext $C_{B,t}$.

3. Transfer transaction stage

When a transaction is involved, the account balance of transaction initiator A is t_A, and the t amount needs to be transferred to transaction receiver B. In the transfer transaction stage, according to the homomorphic encryption blockchain transaction protocol, transaction initiator A first generates a π_1, proving that the t_A input is the transaction amount t, A's own public key pk_A, transaction receiver B's public key pk_B, ciphertext $C_{A,t}$, and ciphertext $C_{B,t}$. When the hash length is $k = 64$, the success probability of fraud verification node V of transaction initiator A generating π_1 is 2-64, that is, the security strength of π_1 reaches 264.

To verify the reasonableness of the above transactions, it is also necessary for transaction initiator A to generate a π_2. The input of π_2 is the pre-transaction A account balance t_A, transaction amount t, A's own public key pk_A, transaction receiver B's public key pk_B, and A's own public key encryption t_A, t, t_{A-t} to generate ciphertext C_{A,t_A}, $C_{A,t}$, $C_{A,tA-t}$ and public parameter D (large enough). In the case of $k = 64$, the probability of the success of transaction initiator A generating π_2 and cheating verification node V is also 2-64, that is, the security strength of π_2 also reaches 264.

In the transfer transaction stage, transaction initiator A transfers t amount to transaction receiver B. In this stage, A takes the public parameters $params$, t, t_A, pk_A, C_{A,t_A}, π_1, π_2, pk_B and the ciphertext $C_{A,tA-t}$ of A's account balance minus the transaction amount $t_A - t$ as input and generates the transfer statement $x = \left(C_{A,t}, C_{B,t}, pk_A, pk_B, C_{A,t_A}, C_{A,t_A-t}\right)$ and the non-interactive zero-knowledge proof $\pi = (\pi_1, \pi_2)$. When A publishes transaction information in the private blockchain, it needs to send x and π to the private blockchain network.

4. Operation phase

Verification node V only needs to verify whether the π provided by transaction initiator A can prove that the transaction is reasonable, that is, it can prove that the transaction amount of transaction initiator A's account decreases, the corresponding transaction amount of transaction receiver B's account needs to be increased, and before trading, trader A's account balance t_A is more than the trading volume t. In this stage, $params$, x, and π are inputs. If the confirmation is correct, there will be an authenticated transaction, which will trigger a smart contract on a blockchain for automatic transfer. The smart contract updates the account balance ciphertext of transaction initiator A to $C'_{A,t_A} = C_{A,t_A} - C_{A,t}$ and the account balance ciphertext of transaction receiver B to $C'_{B,t_A} = C_{B,t_A} - C_{B,t}$ according to the execution result. Otherwise, ignore this transaction. After a period of time, the verified transactions will be confirmed, that is, they will join the blockchain together with the new block. In general, the verification process is carried out on a blockchain. Transaction confirmation can trigger the automatic transfer of smart contracts on the chain.

When a transaction involves a change operation, for example, transaction initiator A transfers t amount to transaction receiver B, in fact, A only needs to transfer t_1 amount, that is, transaction receiver B needs to change $t - t_1$ amount to A. A uses the public key of transaction receiver B to encrypt t as $C_{B,t} = FHE.Enc(pk_B, t)$, and the actual transfer amount t_1 of encrypted transaction is $C_{B,t_1} = FHE.Enc(pk_B, t_1)$. Based on the global privacy protection scheme of homomorphic encryption, other verification nodes V on the blockchain can verify whether the transaction receiver B has completed the zeroing operation.

5. Experiments and Results

The objective of these experiments was to validate the efficacy of the proposed blockchain global privacy protection algorithm based on a digital mutual trust mechanism. The experiments compared the proposed method against two established methods referenced in the literature (hereafter referred to as Method [4] and Method [5]) to evaluate their capabilities in ensuring privacy within and between blockchain networks.

5.1. Experiments Setup

To validate the effectiveness of the proposed blockchain global privacy protection algorithm, a detailed experimental setup was meticulously configured using six high-performance computers as nodes, each equipped with an Intel Core i7 processor, 16 GB RAM, and 512 GB SSD, operating under Linux Ubuntu 20.04. These nodes were interconnected using a gigabit switch within a closed-network environment, and segmented into three logical partitions (global, ns1, and ns2) to facilitate distinct inter- and intra-partition tests. Custom-developed blockchain simulation software, based on the Hyperledger Fabric framework, was modified to incorporate the proposed digital mutual trust mechanism and simulate the reference Methods [4,5]. The standard cryptographic protocols were implemented across all communications within the blockchain network, including the use of TLS/SSL for data transmission security.

5.2. Detection of Inter-Blockchain Privacy Protection Capability

5.2.1. Experimental Scenarios and Methodology

Two primary scenarios were tested to assess the integrity and performance of privacy protection:

Scenario 1: Inter-Blockchain Privacy Protection

Objective: Evaluate how effectively each method protects data privacy across different blockchain partitions.

Process:

Transactions are simulated from the client to various nodes.

Tests are performed to delete and immediately query the partition database of a legal node, testing resilience to data tampering.

Scenario 2: Intra-Blockchain Privacy Protection

Objective: Test the response of the system when dealing with private transactions within the same blockchain partition.

Process:

Simulated private transactions aimed at testing the system's capacity to safeguard transaction confidentiality and data integrity against unauthorized access.

Each scenario was executed 200 times to ensure statistical relevance, and the results were logged for comparative analysis.

The internal data of the partition are maintained by the partition participants. Any malicious node's damage to the data cannot cause damage to the entire partition data. The experiment simulates the data destruction by malicious nodes by deleting the corresponding partitioned database. The specific detection steps are as follows: When the network connection between all nodes is Normal and the connection between the client and all nodes is Normal, the client sends different simulated partition transactions to different nodes, and then simulates the transaction request of the illegal partition. Delete the partition database of a legal node, query the partition data immediately, wait for the node data synchronization to complete, and then query the partition data of the node.

The detection results of inter-blockchain privacy protection by different methods are shown in Table 1.

As demonstrated in Table 1, across 200 experiments, all three methods—the proposed method, as well as reference Methods [4,5]—met the expectations for ensuring the integrity and correctness of privacy data across blockchain chains. Specifically, each method successfully processed and acknowledged all transaction requests sent to Nodes 1 through

4, which are part of the ns1 partition, achieving a perfect success rate of 200/200. Conversely, transactions directed towards Nodes 5 and 6, which lie outside the ns1 partition, were appropriately rejected, as evidenced by the 0/0 success rates for these nodes. This outcome validates the robust partitioning logic inherent in our privacy protection algorithms, effectively safeguarding data integrity and enforcing access controls across different blockchain segments.

Table 1. Detection results of privacy protection correctness between blockchain chains by different methods.

The Reference Method and Scenario	Number of Test Groups	Tested Result Nodes 1 to 4 Successful Sent/Receipt	Nodes 5 Successful Sent/Receipt	Nodes 6 Successful Sent/Receipt
[4]—1	200	200/200	0/0	0/0
[4]—2	200	200/200	-	-
[5]—1	200	200/200	0/0	0/0
[5]—2	200	200/200	-	-

Furthermore, Scenario 2 probes the resilience of our system against threats to data integrity, particularly in situations where a node's database within the ns1 partition is deliberately compromised. The immediate failure of the receipt query following the database deletion, succeeded by a successful query after system recovery, highlights the system's robust capability to restore and synchronize data among partition participants. This mechanism ensures that the integrity and availability of partitioned data are preserved, even amid malicious attempts to disrupt network operations.

Table 1 concisely encapsulates these findings, illustrating a consistent 200/200 success rate for initial transmissions to nodes within the ns1 partition, complemented by effective recovery and data synchronization that affirm the efficacy of our proposed mechanisms.

5.2.2. Performance

In order to further verify the inter-blockchain privacy protection capabilities of the three methods, the single-node performance of the three methods is tested. First, four servers are selected to deploy blockchain nodes, and each node deploys 1~10 blockchain platforms, simulating the scenario where four institutions participate in 1~10 businesses at the same time, and a test machine simulates sending 1~10 business transaction requests to blockchain nodes. The experiment is divided into 10 groups. In each group of experiments, the client's pressure requests are equally distributed to each partition. Then, select the same four servers to deploy blockchain nodes that add the feature of partition consensus. Each node participates in 1~10 partitions, respectively. Simulate the scenario where four institutions participate in 1~10 businesses at the same time. The same test machine simulates sending 1~10 partition transaction requests to blockchain nodes. The experiment is also divided into 10 groups. In each group of experiments, the client's pressure requests are equally distributed to each partition. The total performance value of single-node processing is compared, as shown in Table 2.

Table 2. Single-node performance test results.

Method	Single-Node Processing Total Performance Value									
	1	2	3	4	5	6	7	8	9	10
The proposed method	6005	5588	5861	5752	5703	5626	5606	5218	5019	4830
The reference [4] method	7016	6875	6758	6712	6689	6401	6270	5891	5559	5344
The reference [5] method	6259	6986	6972	6815	6635	6356	6014	5509	5428	5250

As shown in Table 2, no matter how large the partition involved in a node is, compared with Method [4] and Method [5], the method of the invention has the overall processing efficiency value of a single node, which can greatly reduce network resource consumption, that is, the proposed method can bring considerable performance improvement while ensuring correctness and integrity.

5.3. Detection of Privacy Protection Capability in the Blockchain

5.3.1. Experimental Scenarios and Methodology

The objective of these experiments was to evaluate the privacy protection efficacy of the proposed blockchain method in comparison with reference Methods [4,5]. Specifically, the experiments were designed to test the robustness of privacy controls in scenarios involving private transactions within a simulated blockchain environment. Each experiment was designed to test data integrity and privacy enforcement across different blockchain partitions.

Three distinct scenarios were executed to assess the correctness of privacy protection mechanisms under various conditions:

Scenario 1: A private transaction request was sent to Node 1, with hash values collected from Nodes 2 and 3. The expected result was that the privacy transaction request would fail, demonstrating effective access control.

Scenario 2: A private transaction request was sent to Node 1, with hash values collected again from Nodes 2 and 3. Subsequently, the privacy transaction receipt was queried from Nodes 4, 5, and 6. The expected result was that the transaction request would succeed, but the receipt query would fail, indicating restricted data access.

Scenario 3: A private transaction request was sent to Node 1, with hash values collected from Nodes 1 and 2. After deleting the partitioned ledger database of Node 1, an immediate query for the receipt of the private transaction was made, followed by a second query after some time. The expected result was that the initial query would fail but the subsequent query would succeed, testing the resilience and recovery capabilities of the blockchain system.

The correctness detection results of privacy protection in the blockchain chain of different methods are shown in Table 3.

Table 3. Detection results of privacy protection correctness in blockchain chain by different methods.

No	Number of Test Groups	Meet the Expected Number of Groups		
		The Proposed Method	The Reference [4] Method	The Reference [5] Method
1	200	200	200	200
2	200	200	148	153
3	200	200	162	141

The results (Table 3) indicate that in all 200 tests, the proposed method consistently met the expected outcomes, demonstrating superior performance in maintaining transaction privacy compared to the reference methods. Specifically, the proposed method successfully restricted unauthorized access and maintained data integrity after simulated data tampering, highlighting its robustness against privacy breaches. In contrast, reference Methods [4,5] showed lesser efficacy, particularly in scenarios requiring fine-grained control over transaction privacy.

5.3.2. Performance

In blockchain systems, privacy protection algorithms primarily address users' needs for the confidentiality of personal information, with a significant focus on the latency associated with processing private transactions. This experiment was conducted using a network of six nodes configured to simulate various user scenarios to evaluate the performance of our proposed privacy protection algorithm compared to reference Methods [4,5]. The results are shown in Figures 2 and 3:

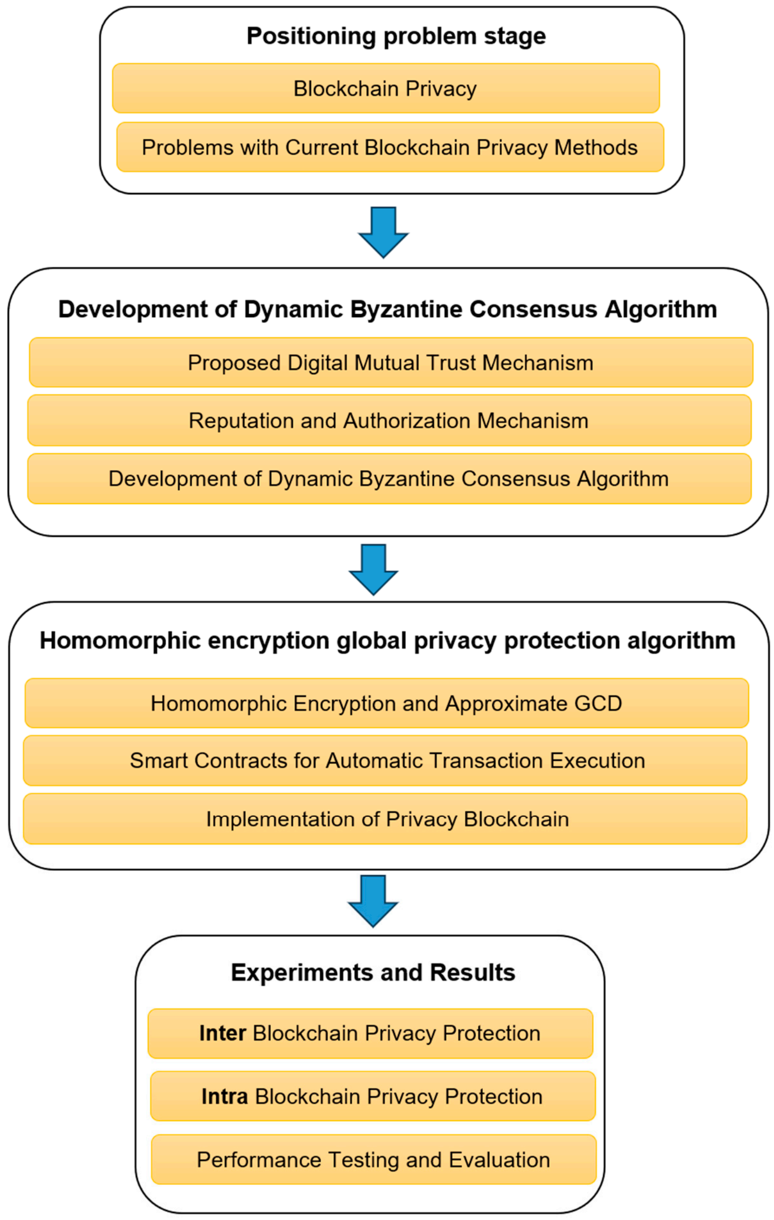
Positioning problem stage
Blockchain Privacy
Problems with Current Blockchain Privacy Methods
Development of Dynamic Byzantine Consensus Algorithm
Proposed Digital Mutual Trust Mechanism
Reputation and Authorization Mechanism
Development of Dynamic Byzantine Consensus Algorithm
Homomorphic encryption global privacy protection algorithm
Homomorphic Encryption and Approximate GCD
Smart Contracts for Automatic Transaction Execution
Implementation of Privacy Blockchain
Experiments and Results
Inter Blockchain Privacy Protection
Intra Blockchain Privacy Protection
Performance Testing and Evaluation

Figure 2. The methodology flow of blockchain privacy with digital mutual trust mechanism.

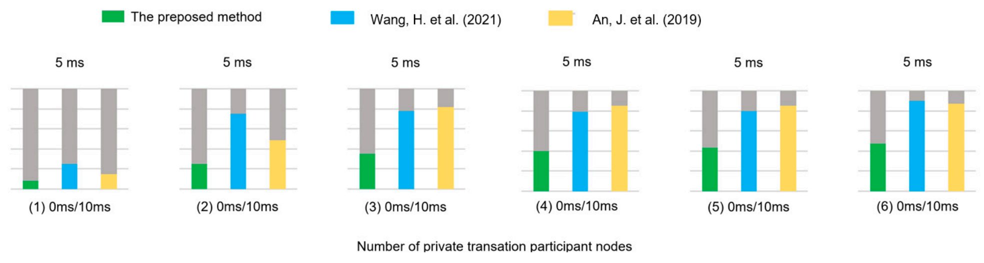

Figure 3. Privacy request delay detection results [4,5].

It can be seen from Figure 3 that when the participants are added one by one, the transfer node needs to synchronize privacy affairs with other privacy participant nodes and wait for other privacy participants' node signature verification, local persistence, and return of a confirmation message, so the delay shows an upward trend; however, the delay of the method of the invention is always shorter than that of the methods referred to in references [4,5], its stability is 5 ms, and the overall delay time is acceptable to users.

As can be seen from Figure 4, as the number of participants increases, the receiving node needs to request the receipt information from other participant nodes, in addition to querying the local database itself, and wait for the local query and return of query results from other privacy participant nodes. Therefore, the delay gradually increases. However, in the whole process of the experiment, the delay of the method is always shorter than that of Method [4] and Method [5], both of which are 2.5 milliseconds, which can meet users' requirements for privacy.

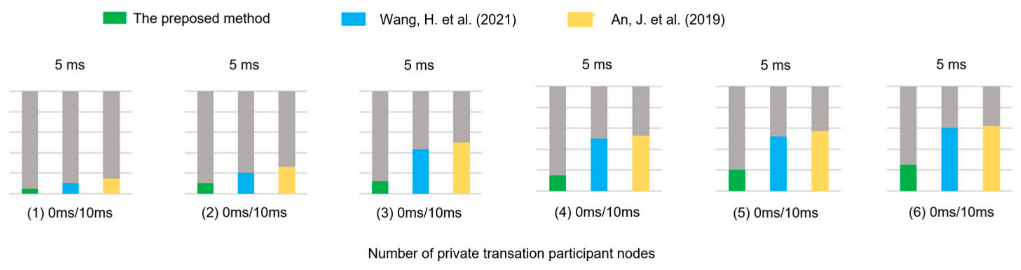

Figure 4. Detection results of privacy transaction receipt query delay [4,5].

The experimental results underscore the superior performance of the proposed method, which integrates a homomorphic encryption blockchain privacy protection algorithm based on smart contracts. This integration not only secures the input, output, and transaction details but also enhances the overall security framework of the blockchain. While reference Methods [4,5] effectively protect data privacy during transmission and ensure data accuracy and integrity, our proposed method excels in reducing operational delays and enhancing transaction efficiency. This capability makes it particularly suitable for applications where timely data processing is crucial, such as in financial transactions or sensitive data exchanges in healthcare systems.

6. Discussion

6.1. Efficiency Considerations in Homomorphic Encryption

The proposed utilization of homomorphic encryption in this study prompts a thorough examination of its efficiency, particularly concerning computation time and calculation operations. Homomorphic operations are categorized into three types: partially homomorphic addition, partially homomorphic multiplication, and fully homomorphic addition and multiplication. Analyzing the application of homomorphic algorithms reveals several critical considerations.

Homomorphic encryption and decryption times are directly proportional to the size of the dataset, with encryption and decryption times typically increasing linearly with the quantity of data. Similarly, the duration for partially homomorphic addition is influenced by both the number of operations and the size of the dataset. Generally, the time for partially homomorphic addition increases proportionally with the growth in dataset size and operation count. Additionally, the computational performance of partially homomorphic multiplication is linearly related to the quantity of data. As the dataset expands, the time required for computation also increases. Furthermore, the performance of partially homomorphic multiplication is non-linearly correlated with the number of operations, with computation time growing by a factor of N as the number of operations increases.

Fully homomorphic operations impose high demands on performance and may not be suitable for all scenarios. For real-time tasks with small dataset sizes, it is advisable to solely consider partially homomorphic addition. Conversely, for non-real-time tasks, it is preferable to avoid using partially homomorphic multiplication operations and instead opt for partially homomorphic addition operations to complete homomorphic calculations.

6.2. Limitations of Nodes in Blockchain Networks

In addition to exploring the efficiency implications of employing homomorphic encryption within blockchain networks, it is imperative to delve into the intricacies of the nodes operating within these decentralized systems. In the realm of blockchain and cryptography, nodes serve as the fundamental building blocks, each maintaining a copy of the distributed ledger and participating in transaction validation and consensus mechanisms. However, these nodes are not without their limitations, which can significantly influence the overall performance and scalability of the network.

One critical limitation lies in the computational power of individual nodes. The cryptographic operations involved in homomorphic encryption, such as encryption, decryption, and homomorphic computations, require substantial computational resources. Nodes with limited computing capabilities may struggle to perform these operations efficiently, leading to potential bottlenecks in transaction processing and network throughput. Moreover, as the blockchain network grows in size and complexity, the computational demands placed on each node increase proportionally, exacerbating these limitations further.

Another factor to consider is the storage capacity of nodes within the blockchain network. The decentralized nature of the blockchain necessitates that each node stores a copy of the entire transaction history, known as the blockchain ledger. As the volume of transactions grows over time, so does the size of the blockchain, placing strain on the storage capacity of individual nodes. Nodes with insufficient storage may face challenges in maintaining a complete and up-to-date copy of the blockchain, potentially compromising the integrity and security of the network.

Furthermore, bandwidth constraints can pose significant challenges for nodes, particularly in blockchain networks where peer-to-peer communication is prevalent. Nodes rely on network connectivity to broadcast transactions, propagate blocks, and participate in consensus protocols. Limited bandwidth can result in delays in transaction propagation, increased latency in block propagation, and potential forking issues within the network. Consequently, nodes operating in environments with constrained bandwidths may struggle to keep pace with the demands of the blockchain network, impacting overall network performance and reliability.

6.3. Key Size and Validation in Encryption and Decryption

Validation against the encryption key is a cornerstone of cryptographic security, particularly in the context of homomorphic encryption within blockchain systems. This process is pivotal in verifying that encrypted data remain intact and confidential throughout their lifecycle. Essentially, it ensures that only authorized parties with access to the decryption key can decipher the encrypted information accurately. In the realm of blockchain and cryptography, the integrity of encrypted data is paramount, as any compromise could undermine the trust and reliability of the entire network.

The key size used in encryption and decryption plays a pivotal role in determining the robustness of cryptographic security. In cryptographic protocols, such as homomorphic encryption, the key size directly impacts the complexity of encryption algorithms and the computational effort required for decryption. Generally, larger key sizes bolster the security of encrypted data by increasing the complexity of brute-force attacks and other cryptographic vulnerabilities. However, this enhanced security comes at the cost of heightened computational overhead, as larger keys necessitate more extensive computational resources and longer processing times for encryption and decryption operations. Balancing security requirements with computational efficiency is a critical consideration when determining the appropriate key size for encryption and decryption operations within blockchain networks. While larger key sizes offer greater security assurances, they may impose significant performance penalties on resource-constrained nodes, hindering the overall efficiency and scalability of the network. Conversely, smaller key sizes may offer improved computational efficiency but could compromise the security of encrypted data, rendering them more susceptible to cryptographic attacks.

Achieving an optimal balance between security and efficiency requires a nuanced understanding of the cryptographic algorithms employed, the computational capabilities of network nodes, and the specific security requirements of the blockchain application. Moreover, advancements in cryptographic research and algorithmic optimizations play a crucial role in mitigating the trade-offs between security and efficiency, enabling the development of more robust and scalable cryptographic solutions tailored to the unique challenges of blockchain environments.

7. Conclusions

In public blockchain networks, the transparency of transaction data poses significant privacy risks by enabling the potential tracing of transaction pathways and identification of users, thus endangering privacy. This highlights the necessity for robust blockchain data encryption to effectively protect privacy rights. This study introduces a novel blockchain privacy protection algorithm based on a digital mutual trust mechanism. It enhances the Practical Byzantine Fault Tolerance (PBFT) consensus algorithm with a reputation model and authorization mechanism, leading to a dynamic Byzantine consensus algorithm. This serves as the foundation for our privacy strategy, further strengthened by advancing homomorphic encryption using the approximate greatest common divisor technique. The integration of smart contracts enables automated transaction executions, establishing a comprehensive privacy protection framework. From experimental results, it reduces delay times for privacy requests by 5.3% compared to traditional methods. This approach ensures blockchain technology's secure application, meeting stringent privacy requirements. Our work contributes to advancing blockchain privacy protection discourse, offering a scalable and effective solution to a key sector challenge.

Author Contributions: Conceptualization, Z.C.; methodology, S.P.; software, W.L.; validation, S.H.; formal analysis, W.L.; investigation, Z.C.; resources, S.P.; data curation, S.P.; supervision, L.Z.; project administration, L.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Guangdong Philosophy and Social Science Planning Project (GD23SQGL03); Zhuhai Industry University Research Cooperation and Basic and Applied Basic Research Project in 2020: Research on Key Technologies of Cross-domain Data Compliance and Mutual Trust Computing in Zhuhai and Macau (No. ZH22017002200011PWC); and Science Research Project of Hebei Education Department (BJK2024111).

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: Authors Sheng Peng was employed by the company Zhuhai Yingying Technology Co., Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest. The Zhuhai Yingying Technology Co., Ltd. had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Chudzyński, F.; Struzik, Z. Criticality of Bitcoin Market. *Acta Phys. Pol. A* **2021**, *139*, 447–450. [CrossRef]
2. Kim, H.M.; Laskowski, M.; Zargham, M.; Turesson, H.; Barlin, M.; Kabanov, D. Token Economics in Real Life: Cryptocurrency and Incentives Design for Insolar's Blockchain Network. *Computer* **2021**, *54*, 70–80. [CrossRef]
3. Jin, X.; Chen, P.Y.; Hsu, C.Y.; Yu, C.M.; Chen, T. CAFE: Catastrophic data leakage in vertical federated learning. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 994–1006. [CrossRef]
4. Wang, H.; Liao, J. Blockchain Privacy Protection Algorithm Based on Pedersen Commitment and Zero-knowledge Proof. In Proceedings of the 2021 4th International Conference on Blockchain Technology and Applications, Xi'an, China, 17–19 December 2021; pp. 1–5. [CrossRef]
5. An, J.; Yang, H.; Gui, X.; Zhang, W.; Gui, R.; Kang, J. TCNS: Node selection with privacy protection in crowdsensing based on twice consensuses of blockchain. *IEEE Trans. Netw. Serv. Manag.* **2019**, *16*, 1255–1267. [CrossRef]
6. Garcia, R.D.; Ramachandran, G.S.; Jurdak, R.; Ueyama, J. A Blockchain-based Data Governance Framework with Privacy Pro-tection and Provenance for e-Prescription. *arXiv* **2021**, arXiv:2112.13956. [CrossRef]

7. Steffen, S.; Bichsel, B.; Baumgartner, R.; Vechev, M. ZeeStar: Private Smart Contracts by Homomorphic Encryption and Ze-ro-knowledge Proofs. In Proceedings of the 2022 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 23–25 May 2022; pp. 179–197. [CrossRef]
8. Chen, J.; Li, K.; Yu, P.S. Privacy-Preserving Deep Learning Model for Decentralized VANETs Using Fully Homomorphic En-cryption and Blockchain. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 11633–11642. [CrossRef]
9. Ali, A. A Novel Homomorphic Encryption and Consortium Blockchain-Based Hybrid Deep Learning Model for Indus-trial Internet of Medical Things. *IEEE Trans. Netw. Sci. Eng.* **2023**, *10*, 2402–2418. [CrossRef]
10. Wu, X.; Yu, F.; Wang, J.; Chang, J.; Feng, X. Bpf-payment: Fair payment for cloud computing with privacy based on blockchain and homo-morphic encryption. *Peer-to-Peer Netw. Appl.* **2023**, *16*, 2649–2666. [CrossRef]
11. Jia, B.; Zhang, X.; Liu, J.; Zhang, Y.; Huang, K.; Liang, Y. Blockchain-Enabled Federated Learning Data Protection Aggregation Scheme With Differential Privacy and Homomorphic Encryption in IIoT. *IEEE Trans. Ind. Inform.* **2022**, *18*, 4049–4058. [CrossRef]
12. Tang, F.; Ling, G.; Cai, C.; Shan, J.; Liu, X.; Tang, P.; Qiu, W. Solving Small Exponential ECDLP in EC-Based Additively Homomorphic Encryption and Applications. *IEEE Trans. Inf. Forensics Secur.* **2023**, *18*, 3517–3530. [CrossRef]
13. Mišić, J.; Mišić, V.B.; Chang, X.; Qushtom, H. Adapting PBFT for use with blockchain-enabled IoT systems. *IEEE Trans. Veh. Technol.* **2020**, *70*, 33–48. [CrossRef]
14. Do, T.; Nguyen, T.; Pham, H. Delegated proof of reputation: A novel blockchain consensus. In Proceedings of the 2019 Inter-national Electronics Communication Conference, Okinawa, Japan, 7–9 July 2019; pp. 90–98. [CrossRef]
15. Liu, Y.; Xu, G. Fixed degree of decentralization DPoS consensus mechanism in blockchain based on adjacency vote and the average fuzziness of vague value. *Comput. Netw.* **2021**, *199*, 108432. [CrossRef]
16. Maskey, S.; Badsha, S.; Sengupta, S.; Khalil, I. Reputation-based miner node selection in blockchain-based vehicular edge computing. *IEEE Consum. Electron. Mag.* **2020**, *10*, 14–22. [CrossRef]
17. Jin, Z.; Qiao, Y. A novel node selection scheme for energy-efficient cooperative spectrum sensing using D–S theory. *Wirel. Netw.* **2020**, *26*, 269–281. [CrossRef]
18. Gu, S.; Wang, Y.; Wang, Y.; Zhang, Q.; Qin, X. Grouping-based consistency protocol design for end-edge-cloud hierarchical storage system. *IEEE Access* **2020**, *8*, 8959–8973. [CrossRef]
19. Kwon, T.; Ju, H.; Lee, H. Performance study for random access-based wireless mutual broadcast networks with ginibre point processes. *IEEE Commun. Lett.* **2020**, *24*, 1581–1585. [CrossRef]
20. Winter, D.K.; Khatri, R.; Schmidt, M. Decentralized prosumer-centric P2P electricity market coordination with grid security. *Energies* **2021**, *14*, 4665. [CrossRef]
21. Ye, T.; Ding, J.; Lee, T.T.; Maier, G. AWG-Based Nonblocking Shuffle-Exchange Networks. *IEEE/ACM Trans. Netw.* **2020**, *28*, 2699–2712. [CrossRef]
22. Kocanaogullari, D.; Eksioglu, E.M. Deep learning for MRI reconstruction using a novel projection based cascaded network. In Proceedings of the 2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP), Pittsburgh, PA, USA, 13–16 October 2019. [CrossRef]
23. Drucker, N.; Gueron, S. Combining homomorphic encryption with trusted execution environment: A demonstration with paillier encryption and SGX. In Proceedings of the International Workshop on Managing Insider Security Threats ACM 2017, Dallas, TX, USA, 30 October 2017. [CrossRef]
24. Zhu, H.; Wang, C.; Wang, X. Quantum fully homomorphic encryption scheme for cloud privacy data based on quantum circuit. *Int. J. Theor. Phys.* **2021**, *60*, 2961–2975. [CrossRef]
25. Almazaydeh, L. Secure RGB image steganography based on modified LSB substitution. *Int. J. Embed. Syst.* **2020**, *12*, 453–457. [CrossRef]
26. Chen, Y.; Hu, B.; Yu, H.; Duan, Z.; Huang, J. A threshold proxy re-encryption scheme for secure IoT data sharing based on blockchain. *Electronics* **2021**, *10*, 2359. [CrossRef]
27. Bourne, M.; Winkler, J.R.; Su, Y. The computation of the degree of the greatest common divisor of three Bernstein basis polynomials. *J. Comput. Appl. Math.* **2019**, *373*, 112373. [CrossRef]
28. Wu, L. Full-homomorphic encryption simulation of character data under active network defense. *Comput. Simul.* **2019**, *36*, 289–292. [CrossRef]
29. Wang, Z.; Jin, H.; Dai, W.; Choo, K.K.R.; Zou, D. Ethereum smart contract security research: Survey and future research op-portunities. *Front. Comput. Sci.* **2021**, *15*, 152802. [CrossRef]
30. Huang, Y.; Wang, B.; Wang, Y. Research and application of smart contract based on ethereum blockchain. *J. Phys. Conf. Ser.* **2021**, *1748*, 042016. [CrossRef]
31. Wang, J.; Xu, S.; Zheng, F.; Lu, K.; Song, J.; Shao, L. Learning efficient hash codes for fast graph-based data similarity retrieval. *IEEE Trans. Image Process.* **2021**, *30*, 6321–6334. [CrossRef]