applied sciences

MDPI

Article

Blockchain-Based Distributed Computing Consistency Verification for IoT Mobile Applications

Jiahao Zhao, Yushu Zhang * and Jiajia Jiang

College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China; jiahaozhao@nuaa.edu.cn (J.Z.); jiangjiajia@nuaa.edu.cn (J.J.)
* Correspondence: yushu@nuaa.edu.cn

Abstract: The maturation of wireless connectivity, blockchain (distributed ledger technologies), and intelligent systems has fostered a comprehensive ecosystem for the Internet of Things (IoT). However, the growing volume of data generated by IoT devices creates substantial pressure on blockchain storage and computation capabilities, impeding the further development of the IoT ecosystem. Decentralizing data storage across multiple chains and utilizing cross-chain technology for data exchange eliminates the need for expensive centralized infrastructure, lowers data transfer costs, and improves accessibility. Hence, the issue of computational and storage pressure in blockchain can be improved. Nonetheless, the data of IoT devices are constantly updating, and ensuring consistency for dynamic data across heterogeneous chains remains a significant challenge. To address the aforementioned challenge, we propose a blockchain-based distributed and lightweight data consistency verification model (BDCA), which leverages a batch verification dynamic Merkle hash tree (BV-MHT) and an advanced gamma multi-signature scheme (AGMS) to enable consistent verification of dynamic data while ensuring secure and private data transmission. The AGMS scheme is reliable and robust based on security analysis while the dependability and consistency of BDCA are verified through inductive reasoning. Experimental results indicate that BDCA outperforms CPVPA and Fortress in communication and computation overhead for data preprocessing and auditing in a similar condition, and the AGMS scheme exhibits superior performance when compared to other widely adopted multi-signature schemes such as Cosi, BLS, and RSA. Furthermore, BDCA provides up to 99% data consistency guarantees, demonstrating its practicality.

Keywords: IoT mobile applications; cross-chain; data consistency verification; data dynamism

Citation: Zhao, J.; Zhang, Y.; Jiang, J. Blockchain-Based Distributed Computing Consistency Verification for IoT Mobile Applications. *Appl. Sci.* **2023**, *13*, 7762. https://doi.org/10.3390/app13137762

Academic Editors: Adnan Anwar, Walayat Hussain, Mian Ahmad Jan and Syed Rooh Ullah Jan

Received: 6 June 2023
Revised: 26 June 2023
Accepted: 29 June 2023
Published: 30 June 2023

1. Introduction

As a state-of-the-art technology, the Internet of Things (IoT) has garnered extensive application in various facets of modern life, ranging from smart cities [1] and connected vehicles [2] to intelligent healthcare [3–5]. The proliferation of IoT devices has resulted in the generation of voluminous and dynamic data [6], which often contains sensitive user information, including personal identification, health, physiological data, and transaction records [7]. The exponential growth of data has placed significant pressure on the storage and computing capabilities of IoT devices. Moreover, due to the problem of data silos [8], secure and reliable data interaction among IoT devices remains a challenging issue, thereby impeding the further development of IoT.

Currently, the prevalent approach to address the storage and computation pressure in the IoT domain is to store data in cloud servers [9–13], which possess robust computational and storage capabilities. Notwithstanding the potential benefits of cloud-based solutions for IoT data storage, the adoption of this approach also introduces new challenges. Firstly, the process of uploading data to cloud servers may result in delays or service disruptions for IoT devices, particularly in the presence of external attacks that specifically target cloud servers. Secondly, uploading data to cloud servers relinquishes the control of data

owner over their data, leading to potential data inconsistency and security breaches. As a result, data may be subjected to various issues, such as theft, tampering, or forgery, thereby compromising its integrity and confidentiality. Despite the best efforts of cloud service providers to maintain data consistency, data loss events still occur frequently. For instance, in 2015, a Google Cloud server data centre was attacked, resulting in the permanent loss of a significant amount of user data. Similarly, in 2016, Uber experienced an external attack that resulted in the leakage of around 57 million user records. These events underscore the fact that data loss or leakage due to cloud service providers is a common occurrence [14]. Thus, ensuring secure and reliable data storage remains a critical research challenge.

The emergence of blockchain technology provides a promising alternative for addressing these challenges [15,16]. Blockchain possesses several essential characteristics, including transparency, tampering, and decentralization, which enable distributed data storage and consistency verification and ensure the security and privacy of data. Additionally, cross-chain technologies can facilitate data exchange and interaction among different IoT devices [17–19], thereby mitigating the problem of data silos and promoting further development in the IoT domain. However, ensuring secure and reliable data storage as well as achieving traceability and consistency verification of data interactions remain critical research challenges. Currently, the focus of research on addressing data consistency auditing primarily centres on traditional cloud storage. Provable data possession (PDP) [20] is a key technique for verifying data consistency, which utilizes a third-party auditor to conduct audits on the data. However, the reliability of third-party auditors cannot be guaranteed, and this approach undermines the decentralized feature of blockchain technology.

To address the above-mentioned problems, in this paper, we propose a novel blockchain-based distributed and lightweight data consistency verification model (BDCA), which ensures the consistency of data transmission during cross-chain interactions. The BDCA model comprises four entities: IoT device (DO), source chain (SC), target chain (TC), and audit chain (AC). AC is a distinct chain that audits data consistency during cross-chain interactions between SC and TC. The audit information is recorded in AC, which is transparent, tamper-proof, and preserves the decentralized feature of blockchain technology. To account for the possibility of data modification in the blockchain, we design a batch verification dynamic Merkle hash tree (BV-MHT), which stores the position information of the data to facilitate data location. Additionally, we utilize the auxiliary verification information form (AVF) to implement data updating and auditing with minimal overhead. Furthermore, in order to ensure the security and privacy of the transferred data, we introduce an advanced gamma multi-signature scheme (AGMS) to encrypt the exchange data between SC and TC. The main contributions of this paper are as follows:

- We propose a novel blockchain-based distributed and lightweight data consistency verification model (BDCA), which provides robust data security and privacy guarantees. Furthermore, we demonstrate that BDCA can effectively achieve cross-chain data interaction consistency verification with minimal overhead through experimental evaluations.
- We design a batch verification dynamic Merkle hash tree (BV-MHT), which supports batch verification and dynamic data updates. Furthermore, we introduce the concept of the auxiliary verification information form (AVF) to locate the position of the stored data and improve batch validation efficiency.
- We construct random challenges and store the audit logs in the audit chain (AC) for future checking, enabling efficient and reliable cross-chain data consistency verification while preserving the transparency and tamper-proof feature of the blockchain.

The remainder of the paper is organized as follows. First, we summarize the related work in Section 2 and introduce preliminaries of BDCA in Section 3. Second, we format BDCA in Sections 4 and 6. Third, we provide the security analysis and theoretically and experimentally analysis of BDCA in Sections 7 and 8, respectively. Lastly, we summarize the paper in Section 9.

2. Related Work

2.1. Cross-Chain Technologies and Applications

With the development of cutting-edge information technology, ensuring the secure and private flow and storage of data has become an increasingly important concern. In this context, the emergence of cross-chain technology has offered a promising alternative for resolving these challenges. Currently, the mainstream cross-chain technologies include notary mechanisms [17], side/relay chain technology [18], and hash locking [19]. Cross-chain technologies have been utilized in many fields, which facilitates the further development of blockchain.

In 2019, Jiang et al. [21] proposed a cross-chain framework, which integrated multi-chains to implement efficient and secure IoT data management. In 2021, Tian et al. [22] proposed a distributed cryptocurrency trading scheme, which can implement secure and fair trading between different types of cryptocurrencies. In 2022, Xiong et al. [23] proposed a notary group-based cross-chain interaction model, which can achieve efficient interoperability between different blockchains. In 2022, Herlihy et al. [24] proposed a novel concept "cross-chain deal", a new approach to managing the complex distributed computations and assets flow in an adversarial setting. In 2022, Li et al. [25] proposed a privacy-preserving cross-chain solution based on sidechain, which can guarantee transaction unlinkability, exchange fairness, and value confidentiality. Nevertheless, these studies mainly focused on static data flow or asset transfer while dynamic data flow and data consistency verification lack sufficient research.

2.2. Dynamic Data Integrity Auditing Scheme

Currently, dynamic data integrity auditing schemes are primarily focused on cloud scenarios. In 2007, Ateniese et al. [20] firstly proposed a provable data possession (PDP) scheme, which can implement the integrity auditing of data with high probability. In 2008, Ateniese et al. [26] improved the traditional PDP scheme and proposed a partially dynamic PDP scheme, which can implement the insertion, deletion, and modification of file blocks with a restricted limit. From then on, many kinds of research implementing dynamic data integrity auditing as well as supporting various properties in different fields have been proposed. In 2015, Tian et al. [27] proposed a privacy preservation data integrity auditing scheme, which enables efficient data updating. In 2017, Rao et al. [28] proposed a data integrity auditing scheme, which supports fully dynamic data updating and can detect malicious data owners or dishonest behaviours. In 2017, Shen et al. [29] proposed an efficient public auditing protocol, which can implement global and sampling blockless verification as well as batch auditing, utilizing a doubly linked info table and a location array. However, these studies introduce a third-party auditor (TPA), which may disrupt the decentralized feature of the blockchain and the reliability of the TPA is also not assured.

2.3. Gamma Signature Scheme

Gamma signature, originally proposed by Yao et al. [30] in 2013 as a modification of Schnorr signature [31], is characterized by a two-phase implementation: an offline phase that pre-computes partial values without any knowledge of the message to be signed and an online phase that generates the aggregated signature upon receiving the message. Compared to Schnorr signature, Gamma signature offers several advantages, including superior online performance, greater flexibility in interactive protocols, and enhanced unforgeability against concurrent interactive attacks. The advanced gamma multi-signature scheme (AGMS) is a novel multi-signature scheme based on Gamma signature, improving the efficiency and security of Gamma signature and making its application in blockchain feasible.

2.4. Merkle Hash Tree

Merkle hash tree (MHT), introduced by Merkle [32] in 1989, is a popular technique for verification and integrity checking in various applications, such as Git and Bitcoin, where it serves as an authentication scheme [33]. MHT is a binary tree composed of hash

values, with leaf nodes being arbitrary hash values or those generated from pseudorandom numbers, and intermediate nodes being the hashes of their immediate children. The root of the MHT is a unique value due to the collision resistance property of the hash function, which ensures that no two hash values differing by at least one bit should be the same. To adapt MHT to the IoT environment, traditional designs have been modified. BV-MHT is a variation of the traditional MHT, where the data stored in each node is adjusted to enable efficient data updates as well as batch verification.

3. Preliminaries

3.1. Advanced Gamma Multi-Signature Scheme

The advanced gamma multi-signature scheme (AGMS) is a novel variant of the gamma signature scheme, which leverages the discrete logarithmic puzzle [34] and the efficient Schnorr signature scheme [35] to enable multiple entities to collaboratively generate an aggregated signature based on an aggregated public key. The concrete procedures of the scheme entail a rigorous and well-defined process, which are shown as follows:

- $Setup(1^n)$: take a security parameter n as the input, output a public parameter $pp = (q, \mathcal{G}, g)$, where $\mathcal{G}$ is a group of prime order q and$g$ is a generator of $\mathbb{G}$.
- $KenGen(pp)$: take a public parameter pp as the input, randomly choose $\{sk_{ij}\}_{i\in[1,n],j\in U_i} \in \mathcal{Z}_N$, compute $\{pk_{ij} = g^{sk_{ij}}\}_{i\in[1,n],j\in U_i}$, and output $\{(pk_{ij}, sk_{ij})\}_{i\in[1,n],j\in U_i}$.
- $KeyAgg(\{pk_{ij}\}_{i\in[1,n],j\in U_i})$: take the public key $(\{pk_{ij}\}_{i\in[1,n],j\in U_i})$ as the input, compute the aggregated public key $PK = \prod_{i\in[1,k]} (\prod_{j\in U_i} pk_{ij})$, and output PK.
- $Sign(msg, V^*, PK)$: take the message m, an aggregation commitment V^*, and an aggregation public key PK as the input, and output a signature (c, Sig).
- $Verify(pp, V^*, V^{*'})$: take the message m, aggregated commitment V^*, and $V^{*'}$ as the input, and output $0/1$ to indicate whether the message msg is invalid or valid.

3.2. Bilinear Map

Let $\mathcal{G}_1$, $\mathcal{G}_2$, and $\mathcal{G}_T$ be a multiplicative cyclic additive group with the same order q. The map $e : \mathcal{G}_1 \times \mathcal{G}_2 \to \mathcal{G}_T$ is a bilinear pairing only if the following properties are satisfied:

- Bilinearity: For any $a, b \in Z_q^*$, $x \in \mathcal{G}_1$, and $y \in \mathcal{G}_2$, $e(x^a, y^b) = e(x, y)^{ab}$.
- Non-degeneracy: For any $x \in \mathcal{G}_1$ and $y \in \mathcal{G}_2$, if $e(x, y) = 1$ only if $x = 1$ or $y = 1$.
- Computability: There exists an efficiently computable homomorphism between $\mathcal{G}_1$ and $\mathcal{G}_2$.

3.3. Mathematical Assumptions

Let $\mathcal{G}$ be an additive cyclic group of prime order q and a, b be random elements of $\mathcal{Z}_N$.

- Discrete logarithm (DL) assumption. Given $g, g^a \in \mathcal{G}$ as the input, it is computationally infeasible to compute a. That is, for any PPT adversary $\mathcal{A}$, the probability of solving the DL problem is negligible in $\mathcal{G}$.
- Computational Diffie–Hellman (CDH) assumption. Given $g, g^a, g^b \in \mathcal{G}$ as the input, it is computationally infeasible to compute g^{ab}. That is, for any PPT adversary $\mathcal{A}$, the probability of solving the CDH problem is negligible in $\mathcal{G}$.

4. Models

In this section, we describe the architecture of BDCA in Section 5.1 and then lay out its threat model in Section 5.2. Finally, we introduce the design goals of BDCA in Section 5.3. Furthermore, some additional notations and descriptions are defined in Table 1.

Table 1. Notations and descriptions.

Notations	Descriptions
M	The original data
M^*	The blinded data
$M^{*'}$	The updated data
d	A public key of DO_i
$\{e, u\}$	A private key of DO_i
n	System security parameter
Tx_1	Data uploading transaction
Tx_2	Data updating request transaction
DO_{s_0}	The leader of all the signers in blockchain
DO_S	The IoT device of SC
DO_T	The IoT device of TC
ϖ	The threshold number of signers

5. The Proposed BDCA

5.1. System Model

The architectural diagram depicted in Figure 1 illustrates the configuration of BDCA, comprising three distinct consortium blockchains based on Hyperledger Fabric technology: the source chain (SC), the target chain (TC), and the audit chain (AC). The innovative model BDCA enables the secure and seamless exchange of data between the two different chains while ensuring data integrity through the implementation of an audit chain (AC) for data consistency verification as well as maintaining the decentralized feature of BDCA [36]. To further enhance the reliability and functionality of BDCA, three smart contracts are deployed within the system, each specifically allocated to SC, TC, and AC, respectively. The four principal entities comprising BDCA are IoT devices (DO), source chain (SC), target chain (TC), and audit chain (AC) with each entity bearing unique responsibilities and obligations as follows:

- *IoT device (DO)*: DO is an entity that is responsible for processing and storing the original data M, and initiating data update requests to the source chain (SC). To ensure the security and privacy of the data, DO preprocesses M into blinded data M^* before generating authentication tags for M^*. Once DO has generated the authentication tags for M^*, it transfers the blinded data and corresponding tags to the audit chain (AC) for secure storage. Within the BDCA model, there are two distinct DO entities, namely DO_S and DO_T, which are deployed within the source chain (SC) and target chain (TC), respectively.
- *Source chain (SC)*: SC is an entity that serves as the primary repository for data received from DO and facilitates the exchange of data with the target chain (TC). Upon receiving data update requests from DO_S, SC is responsible for updating the data accordingly and synchronizing the data updates with TC to ensure consistency across the two chains. In addition, SC is programmed to respond to audit challenges issued by the audit chain (AC) by generating an audit proof that verifies the consistency and accuracy of the data stored within the system. Once the audit proof is generated, SC sends it to AC for data consistency verification, ensuring that the data remains secure and reliable.
- *Target chain (TC)*: TC is an entity that is responsible for receiving data from the source chain (SC) and facilitates data exchange between the two chains. Upon receiving data update requests from SC, TC updates the data as required, ensuring that the data remains consistent across both chains. Like SC, TC also responds to audit challenges from the audit chain (AC) by generating storage proofs that verify the integrity and accuracy of the data stored within the system. Once the storage proof is generated, TC sends it to AC for data consistency verification, ensuring that the data remains secure and reliable.
- *Audit chain (AC)*: AC is an entity that is established and overseen by national regulatory authorities. AC is responsible for conducting data consistency verification between the source chain (SC) and the target chain (TC) based on the audit proof and storage

proof received from *SC* and *TC*, respectively. Through its advanced data consistency verification protocols, *AC* ensures that the data exchanged between *SC* and *TC* remains accurate and consistent, preventing any unauthorized modifications or tampering.

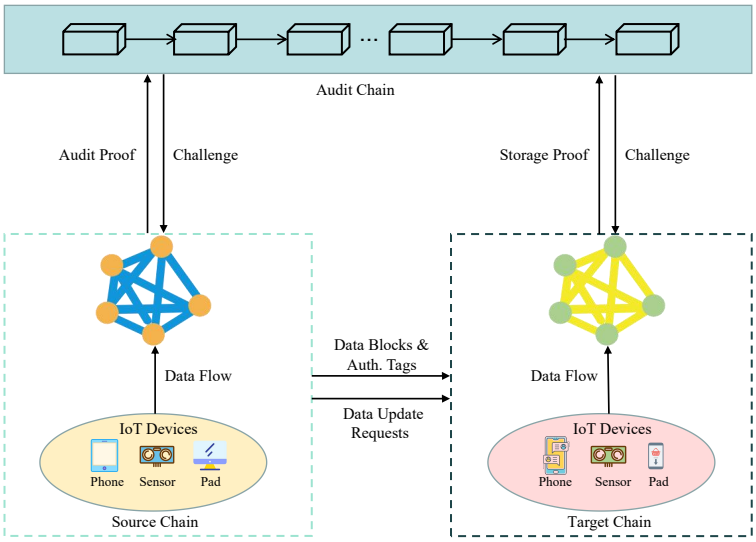


Figure 1. The architecture of BDCA.

5.2. Threat Model

In BDCA, the auditing mechanism is facilitated by *AC* and its associated smart contract, which are deployed by national regulatory authorities. The audit process is transparent and tamper-proof, thereby instilling trust in the reliability of *AC*. Furthermore, it is worth noting that the introduction of the auditing mechanism facilitated by *AC* does not compromise the decentralized feature of the blockchain owing to the openness and transparency of the audit process, ensuring that the regulatory oversight is conducted in a fair and impartial manner. The introduction of *AC* can be viewed as a complementary measure that enhances the security and reliability of cross-chain transactions. Furthermore, *DO* is entrusted with the responsibility of processing the original data and transmitting it to *SC* for storage. However, *SC* is considered semi-honest, which implies that it may have a vested interest in the content of the received data. Similarly, *TC* is also semi-honest and may resort to forging storage proof to bypass the consistency verification process of *AC*. In conclusion, there are mainly the following types of cross-chain attacks:

- *Tampering attacks*: The transferred data $F^{'}$, corresponding tags, and data update requests may be tampered with or forged by an adversary in the process of cross-chain interaction.
- *Privacy leakage attacks*: The content of the transferred data $F^{'}$, corresponding tags, and data update requests may be leaked and expose the private information of *DO* in the process of cross-chain interaction.
- *Audit inconsistency attacks:* The *TC* may pretend to forge a storage proof to pass the data consistency verification of *AC* to conceal its incomplete data storage or updates.

5.3. Design Goals

Based on the above analysis of BDCA, in order to ensure the security and privacy of the transferred data and guarantee the consistency of data interactions between *SC* and *TC*, we propose the following design goals:

- *Consistency*: BDCA can ensure the consistency of the data interaction between *SC* and *TC* and when *TC* modifies or does not update the data as requested, it cannot pass the consistency verification by *AC*.
- *Privacy*: BDCA can ensure the privacy of the transferred data. *SC* and *TC* cannot gain any private information of *DO* in the process of cross-chain data interaction.

- *Dynamic operations*: BDCA can allow *DO* to perform data update operations at will with low overhead, including insertions, deletions, and modifications.
- *Security*: BDCA can ensure that the data updates operations, including insertions, deletions, and modifications, are conducted only by the owner of the data.

6. The Proposed BDCA

In this section, we present a detailed description of the proposed BDCA, highlighting how dynamic data consistency audits can be implemented in the process of cross-chain interaction. In general, BDCA comprises five essential procedures, which include: system initialization, data processing, data uploading, data updating, and data auditing.

6.1. System Initialization

In this phase, *DO* in BDCA generates its public and private key pair and outputs the public parameters of BDCA as follows:

- Upon receiving a security parameter 1^n, each *DO* in BDCA selects four random large prime p, q, $p^{'}$, and $q^{'}$. Meanwhile, *DO* computes RSA modulus $N = pq$ and selects a generator g of $\mathcal{QR}_N$, where $p = 2p^{'} + 1$, $q = 2q^{'} + 1$, and $\mathcal{QR}_N$ is a multiplicative cyclic group of quadratic residues modulo N.
- BDCA then selects a hash function $H_1 : \{0,1\}^* \rightarrow \mathcal{Z}_N$, a pseudo-random permutation $\mathscr{F} : \{0,1\}^{k_1} \times \{0,1\}^{log_2^n} \rightarrow \{0,1\}^{log_2^n}$, and a pseudo-random function (PRF) $\mathscr{A} : \{0,1\}^{k_2} \times \{0,1\}^{log_2^n} \rightarrow \mathcal{Z}_N$.
- BDCA then selects a security random large prime e as the public key and computes the private key d, where $e \cdot d \equiv 1(mod(p^{'}q^{'}))$, u is a security random large prime, and $u \neq d$.
- BDCA then generates a public key and private key pair $\{pk = e, sk = (d, u)\}$ and outputs the public parameters $par = \{g, e, N, H_1, \mathscr{F}, \mathscr{A}, \mathcal{QR}_N\}$.

6.2. Data Processing

In this phase, DO_S first preprocesses the original data M and constructs the BV-MHT and auxiliary verification information form (AVF). Then, DO_S uploads the processed datasets to *SC* for storage, the specific steps are shown as follows:

- Given a original data file $M \in \{0,1\}^*$, DO_S splits M into k blocks, represented as $m_{ii\in[1,k]}$. Note that if the last block is not the same size as the other blocks, DO_S pads 0 at the end of the last block. In order to protect the security and privacy of M, DO_S applies a blind signature technique by creating a blinded version of M, denoted as M^*, with $\{m_i^*\}_{i\in[1,k]} = \{m_i\}_{i\in[1,k]} + H_1(md||a_d)$, where $md \in \{0,1\}^*$ is the unique identification of M, $a_d \in \mathcal{Z}_N$ is a random number, and H_1 is a one-way hash function. Additionally, DO_S calculates a verification value $A_d = g^{a_d}$, where g is a generator of a cyclic group $\mathcal{G}$, and N is the order of $\mathcal{G}$.
- DO_S constructs the batch verification Merkle hash tree (BV-MHT) based on the processed data to facilitate efficient and secure data auditing. In BV-MHT, each node ν stores a tuple (l_ν, r_ν, h_ν), which represents the position information of the node ν in BV-MHT. Specifically, if the node ν is the i-th leaf node τ_i, r_ν is set to 1 and $h_\nu = H_1(m_i^*)$. If the node ν is a non-leaf node χ, r_ν stores the number of the leaf nodes that χ can reach from the left to right and $h_\nu = H_1(r_\nu||h_{left_\chi}||h_{right_\chi})$, where h_{left_χ} and h_{right_χ} denote the left and right child nodes of χ [37]. If the node ν is the left child node of its parent node, l_ν is set to 0; the node ν is the right child node of its parent node, l_ν is set to 1; the node ν is the root of the BV-MHT, l_ν is set to 2. The process of constructing BV-MHT is shown in Figure 2 and the nodes $\{\tau_2, \tau_4, \tau_7\}$ are verified simultaneously.
- DO_S constructs the auxiliary verification information form (AVF) to implement the batch verification and accelerate the process of batch verification. In traditional MHT, the i-th leaf node can only be verified one by one with its siblings on the path from the i-th leaf node to the root. For instance, if the nodes $\{\tau_2, \tau_4, \tau_7\}$ want to be ver-

ified simultaneously, they need to generate the auxiliary verification information $\omega_{\tau_2} = \{\tau_1, \chi_{10}, \chi_{14}\}$, $\omega_{\tau_4} = \{\tau_3, \chi_9, \chi_{14}\}$, and $\omega_{\tau_7} = \{\tau_8, \chi_{11}, \chi_{13}\}$, respectively. The node χ_{14} will be retrieved twice, which will cause more overhead. Furthermore, as the number of verified nodes simultaneously increases, the number of duplicate retrieval nodes increase. From the perspective of saving verification overhead, we introduce a concept auxiliary verification information form (AVF) in BV-MHT, as an example is shown in Table 2, the nodes $\{\tau_2, \tau_4, \tau_7\}$ can be verified simultaneously by retrieving the $\text{AVF}_{r\times t}$, where r is the number of the nodes verified simultaneously, t is the max layer of the leaf nodes in BV-MHT, and *Point* is a point that points to the different line in $\text{AVF}_{r\times t}$. The specific procedures of constructing $\text{AVF}_{r\times t}$ are shown in Algorithm 1. Finally, DO_S invokes Algorithm 1 to generate an $\text{AVF}_{1\times t}(v_k)$ for the last leaf node v_k.

- DO_S generates the authenticated tags of the transferred data $\{m_i^*\}_{i\in[1,k]}$ with Equation (1) and uploads the datasets $DS_1 = \{(m_i^*, \zeta_i)\}_{i\in[1,k]}, \{k, h_{root}\}, \{v_k, \text{AVF}_{1\times t}(v_k)\}\}$ to SC for storage.

$$\zeta = (g^{h_{v_i}} \cdot g^{m_i^*})_{i\in[1,k]}^{d}. \tag{1}$$

- After receiving the datasets DS_1 from DO_S, SC generates a data-uploading transaction Tx_1 and sends Tx_1 to TC:

$$Tx_1 = [\text{“}Upload\text{”}, pk_{DO_S}, DS_1].$$

Algorithm 1 Constructing the auxiliary verification information form ($\text{AVF}_{r\times t}$).

Input: BV-MHT, index sets $\{d_1, d_2, ..., d_r\}(d_1 < d_2 < ... < d_r)$ of the verified leaf nodes.
Output: $\text{AVF}_{r\times t}$.

```
i = 1;
while i <= r do
    LN[i] = v_{d_i};
    i = i + 1;
end while
j = 1, Q = t;
while Q > 0 do
    for i = 1, 2, ..., r do
        ϱ = LN[i];
        if ϱ ≠ NULL then
            if the layer number of ϱ == Q then
                if ϱ exists sibling node κ in current LN[] then
                    AVF[i][j] = "Point_ι" (ι ∈ (1, r]);
                    LN[ι] = NULL;
                else ϱ does not exist sibling node κ in current LN[]
                    AVF[i][j] = κ;
                end if
                LN[i] = the parent node of ϱ;
            else the layer number of ϱ ≠ Q
                AVF[i][j] = NULL;
            end if
        else ϱ == NULL
            AVF[i][j] = NULL;
        end if
    end for
    j = j + 1, Q = Q − 1;
end while
return AVF_{r×t};
```

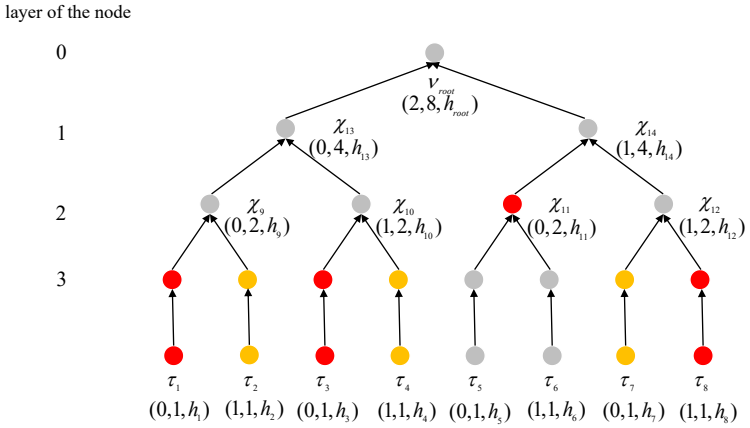


Figure 2. The process of constructing BV-MHT.

Table 2. Auxiliary verification information form (AVF).

$\mathbf{AVI}_{i,j}$	$j=1$	$j=2$	$j=3$
$i=1(Point_1)$	τ_1	$Point_2$	$Point_3$
$i=2(Point_2)$	τ_3	$NULL$	$NULL$
$i=3(Point_3)$	τ_8	χ_{11}	$NULL$

6.3. Data Uploading

In this phase, *SC* encrypts the uploaded transaction Tx_1 with advanced gamma multi-signature scheme (AGMS) to ensure its security and privacy in the process of cross-chain interaction and any unauthorized access or tampering can be detected. The detailed process of AGMS is expounded in Algorithm 2 and the specific process of uploading data is shown as follows:

- *SC* invokes Algorithm 2 to encrypt Tx_1 and obtain the signature (c, sig). Then, *SC* sends the signature (c, sig), Tx_1, and PK_{leader} to *TC*.

Algorithm 2 Advanced gamma multi-signature scheme.

Input: The message msg to be signed, the public key sets of the signers $keys = \{pk_i\}_{i\in[1,\varpi]}$, the threshold number ϖ of signers, and the public key of the leader pk_{leader}.
Output: The multi-signature (c, sig).

1: Randomly select a random value v_i, compute $V_i = g^{v_i}$;
2: Compute the partial aggregated public key $PK_{signers} = \prod_{i\in[1,\varpi]} pk_i$ and the partial aggregated commitment $V_{signers} = \prod_{i\in[1,\varpi]} V_i$;
3: Compute the complete aggregated public key $PK_{leader} = PK_{signers} \cdot pk_{leader}$ and the complete aggregated commitment $V_{leader} = V_{signer} \cdot V_{leader}$;
4: Compute the collective challenge $c = H_1(g, V_{leader}, PK_{leader})$;
5: Compute the hash of the signed message $hv = H_1(msg)$ and the partial aggregated signature $sig_{signer} = \sum_{i\in[1,\varpi]} sig_i = \sum_{i\in[1,\varpi]} v_i \cdot c - hv \cdot d_i$;
6: Compute the signature $sig = sig_{leader} + sig_{signer}$;
7: Output the signature (c, sig)

Upon receiving the signed transaction Tx_1 from *SC*, *TC* proceeds to invoke the designated smart contract S_t to conduct a rigorous verification process to ensure the authenticity and validity of Tx_1 before it is stored on the blockchain.

- *TC* computes $hv = H_1(msg)$ and $\widetilde{V} = (g^{sig} PK_{leader}^{hv})^{c^{-1}}$.

- TC verifies the correctness of the signature (c, sig) with Equation (2). If the equation holds, the signature is valid and TC continues to conduct the following verification; otherwise the signature is invalid, and TC rejects to store Tx_1.

$$c \stackrel{?}{=} H_1(g, \widetilde{V}, PK_{leader}). \tag{2}$$

- TC verifies the correctness of $\{k, h_{root}\}$ by checking whether the hash value $H_1(m_k^*)$ based on the datasets $\{m_i^*\}_{i\in[1,k]}$ is equal to h_{v_k} of the node v_k. If the verification passes, TC continues to conduct the following verification; otherwise, Tx_1 is invalid, and TC rejects to store Tx_1.
- TC invokes Algorithm 3 to verify the correctness of $\text{AVF}_{1\times t}(v_k)$. If the algorithm outputs 0, TC rejects to storage of Tx_1; otherwise, TC stores Tx_1 and returns its acceptance and signature of DO_T. It is worth noting that we assume that $PI := \{(\alpha_i, \beta_i)|\alpha_i, \beta_i \in \mathcal{Z}_N, i \in [1, \vartheta]\}$ is a ϑ size set. For any element $y \in \mathcal{Z}_N$, the set operations '$\pm$' of PI are described as follows:

$$\{PI \pm y\} = \{(PI(\alpha) \pm y), (PI(\beta) \pm y)\},$$

where $PI(\alpha) \pm y := \{(\alpha_i \pm y, \beta_i))\}_{i\in[1,\vartheta]}, PI(\beta) \pm y := \{(\alpha_i, \beta_i \pm y)\}_{i\in[1,\vartheta]}$. Furthermore, the operation $Union(PI_i, PI_j)$ is described as follows:

$$Union(PI_i, PI_j) = \{(\alpha_i, \beta_i), (\alpha_j^{'}, \beta_j^{'})\}_{i\in[1,\vartheta_1], j\in[1,\vartheta_2]},$$

where $PI_i = \{(\alpha_i, \beta_i)\}_{i\in[1,\vartheta_1]}$ and $PI_j = \{(\alpha_j, \beta_j)\}_{j\in[1,\vartheta_2]}$.

6.4. Data Updating

In this phase, when DO_S wants to update the data stored in SC at will, DO_S generates a data updating request req and sends it to SC. SC updates the data as the request req and sends a data update request transaction Tx_2 to TC. TC updates the data as the transaction Tx_2. In general, data update operations include $Insertion(I)$, $Modification(M)$, and $Deletion(D)$. It is imperative to highlight that the process of updating data in this paper is instigated by the user node that initially generated the data. Each data update request is accompanied by a public key signature from the data update user node and the execution of the data update operation is contingent upon its successful verification by a majority of nodes in the blockchain. Specifically, the data update operation must be approved by more than 50% of the user nodes in blockchain. The robust approach of this paper provides a high level of security against potential security threats, such as the malicious random or chosen node or link to a blockchain node deletion and the integrity and security of the blockchain are safeguarded. The specific process is shown as follows:

Modification: Assume that DO_S wants to modify the i-th leaf node into $m_i^{*'}$ at ts_2 (e.g., modify m_4^* into $m_4^{*'}$ in Figure 3b).

- DO_S calculates $H_1(m_i^{*'}) = h_i^*$ and $\zeta_i^* = (g^{h_i^*} \cdot g^{m_i^{*'}})^d$.
- DO_S invokes Algorithm 1 to generate an auxiliary verification information form $\text{AVF}_{1\times t}$.
- DO_S modifies the original node into $v_i^* = \{l_{v_i^*}, r_{v_i^*}, h_i^*\}$ and updates the BV-MHT by recalculating all the nodes on the path from the node v_i^* to the root, generating a new hash root h_{root}^*.
- DO_S generates a data update request $req = \{ts_2, Modify, i, h_i^*, AVF_{1\times t}, h_{root}^*, m_i^*, pk_{DO_S}\}$ and sends it to SC.
- SC verifies the request req with Algorithm 3 and updates the data as the request req, generating a new transaction linked to the original blockchain network. Then, SC generates a data update transaction Tx_2 and encrypts Tx_2 with AGMS in a similar process in Section 6.3 and uploads Tx_2 to TC.

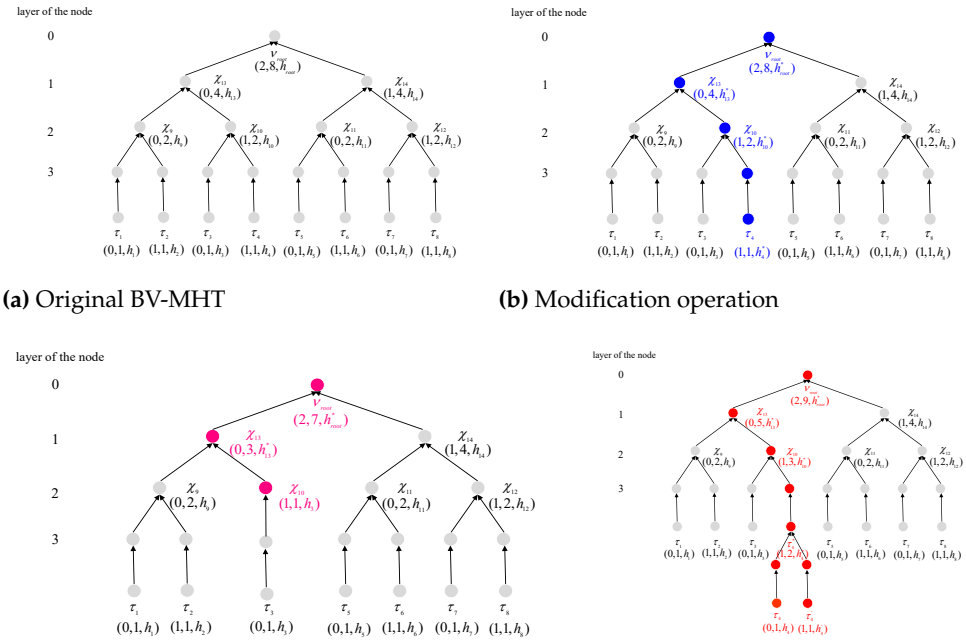
$$Tx_2 = [\text{“}Update\text{”}, req]$$

- After receiving the transaction Tx_2, TC first verifies the correctness and validity of the received transaction in a similar way in Section 6.3. If the verification is valid, TC updates the BV-MHT as requested; otherwise, TC rejects to update the transaction.

Algorithm 3 Batch verification.

Input: Index sets $\{d_1, d_2, ..., d_r\}(d_1 < d_2 < ... < d_r)$ of the verified leaf nodes, the number of all blocks k, the corresponding hash root of BV-MHT based on the batch verified nodes h_{root}, and $\text{AVF}_{r\times t}$.
Output: 0/1.

i = 1;
while i <= r **do**
  $PI_i = \{(k, \gamma_{vd_i} - 1)\}$;
  $\Gamma_i = \gamma_{vd_i}$;
  $\Upsilon_i = h_{vd_i}$;
  i = i + 1;
end while
$Point\_Sets = \{1, 2, ..., r\}$;
for i = 1, 2, ..., t **do**
  for j = 1, 2, ..., r **do**
    if AVF[i][j] == NULL **then**
      continue;
    else if AVF[i][j] is a leaf or non-leaf node v_x **then**
      $\Gamma_i = \Gamma_i + r_{v_x}$;
      if $l_{v_x} == 1$ **then**
        $\Upsilon_i = H_1(\Gamma_i || \Upsilon_i || h_{v_x})$;
        $PI_i = PI_i(\alpha) - r_{v_x}$;
      else if $l_{v_x} == 0$ **then**
        $\Upsilon_i = H_1(\Gamma_i || h_{v_x} || \Upsilon_i)$;
        $PI_i = PI_i(\beta) + r_{v_x}$;
      else $l_{v_x} == 2$
        return 0;
      end if
    else AVF[i][j] is a point $Point_\delta, \delta \in (1, r]$
      $PI_\delta = PI_\delta(\beta) + \Gamma_i$;
      $\Gamma_i = \Gamma_i + \Gamma_\delta$;
      $\Upsilon_i = H_1(\Gamma_i || \Upsilon_i || \Upsilon_\delta)$;
      $PI_i = PI_i(\alpha) - \Gamma_\delta$;
      $PI_i = Union(PI_i, PI_\delta)$;
      remove δ from $Point\_Sets$;
    end if
  end for
end for
if $PI_1 \neq \{(d_1, d_1 - 1), (d_2, d_2 - 1), ..., (d_r, d_r - 1)\}$ **then**
  return 0;
else if $\Gamma_1 \neq k$ or $\Upsilon_1 \neq h_{root}$ **then**
  return 0;
else
  return 0;
end if



(a) Original BV-MHT **(b)** Modification operation

(c) Deletion operation **(d)** Insertion operation

Figure 3. The updating operations of the BV-MHT in BDCA.

Deletion: Assume that DO_S wants to delete the i-th leaf node at ts_2 (e.g., delete m_4^* in Figure 3c).

- DO_S generates the data updates request $req = \{ts_2, delete, i, AFV_{1\times t}, h^*_{root}, pk_{DO_S}\}$ and updates the BV-MHT on the path from the node v_i to the root. Then, DO_S sends it to SC.
- SC verifies and updates the BV-MHT in a similar way as above, and generates a data update transaction Tx_2 encrypted with AGMS. Then, Tx_2 is sent to TC for data updating.

$$Tx_2 = [\text{“}Update\text{”}, req]$$

- TC updates the data in a similar way as above.

Insertion: Assume that DO_S wants to insert a new node after the i-th leaf node at ts_2 (e.g., insert $m_4^{'}$ after m_4 in Figure 3d).

- DO_S generates the data updates request $req = \{ts_2, insert, i, h_i^{'}, m_i^{'}, AFV_{1\times t}, h^*_{root}, pk_{DO_S}\}$ and updates the BV-MHT on the path from the node v_i to the root, where $v_i^{'} = \{l_i, r_i + 1, H_1(r_i + 1||h_i||h_i^{'})\}$. Then, DO_S sends it to SC.
- SC verifies and updates the BV-MHT in a similar way as above, and generates a data update transaction Tx_2 encrypted with AGMS. Then, Tx_2 is sent to TC for data updating.

$$Tx_2 = [\text{“}Update\text{”}, req]$$

- TC updates the data in a similar way as above.

6.5. Data Auditing

In this phase, AC periodically constructs a random challenge to verify the consistency between SC and TC in order to ensure the integrity and reliability of BDCA. It is important to note that all entities involved, including SC, TC, and AC, should utilize a uniform time standard and the time interval for the audit should be fixed. The detailed process is shown as follows:

- *AC* constructs a random challenge:

$$chal^{(ts_1)} = \{ts_1, \lambda_\varphi^{(ts_1)}, \Lambda_\varphi^{(ts_1)}\}_{\varphi\in[1,c]}, \lambda_\varphi^{(ts_1)} \in [1,k], \Lambda_\varphi^{(ts_1)} \in \mathcal{Z}_N \quad (3)$$

where $\{\mathscr{F}_{k_1^{(ts_1)}}(\varphi) = \lambda_\varphi^{(ts_1)}\}_{\varphi\in[1,c]}$ represents the index set of the c challenged block, $\{\mathscr{A}_{k_2^{(ts_1)}}(\varphi) = \Lambda_\varphi^{(ts_1)}\}_{\varphi\in[1,c]}$ denotes the corresponding coefficient set of the challenged block, $k_1^{(ts_1)} = H_2(ts_1||val^{(ts_1)})$, $k_2^{(ts_1)} = H_3(ts_1||non^{(ts_1)})$, H_2 is a hash function that transforms the set $\{0,1\}^*$ to the space of $\mathscr{F}$, H_3 is a hash function that transforms the set $\{0,1\}^*$ to the space of $\mathscr{A}$ [38], $val^{(ts_1)}$ is the number of blocks in TC and $non^{(ts_1)}$ is the random value closest to the nearest the timestamp ts_1. Due to the unpredictability and non-repudiation of $val^{(ts_1)}$ and $non^{(ts_1)}$, the challenge $chal^{(ts_1)}$, which is undeniable, can be constructed by AC.
- *AC* sends the challenge $chal^{(ts_1)}$ to SC and TC.
- Upon receiving the challenge $chal$ from AC, TC selects a secret random value $r^{(ts_1)} \in \mathcal{Z}_N$ and generates a storage proof:

$$proof_{TC} = \{\zeta^{(ts_1)}, \mu^{(ts_1)}, g^{(ts_1)}, u^{(ts_1)}\},$$

where $\zeta^{(ts_1)} = \prod_{\varphi\in[1,c]} \zeta_{\lambda_\varphi^{(ts_1)}}^{\Lambda_\varphi^{(ts_1)}} (modN)$, $\mu^{(ts_1)} = r^{(ts_1)} + \sum_{\varphi\in[1,c]} \Lambda_\varphi^{(ts_1)} \cdot m_{\lambda_\varphi^{(ts_1)}}$, $g^{(ts_1)} = g^{r^{(ts_1)}}$, $u^{(ts_1)} = g_u \sum_{\varphi\in[1,c]} \Lambda_\varphi^{(ts_1)} \cdot m_{\lambda_\varphi}^{(ts_1)} modN$, and $g_u = g^u$.
- TC sends the proof $proof_{TC}$ and its signature pk_{DO_T} used to provide non-repudiation to AC for consistency verification.
- SC generates an audit proof and sends the proof $proof_{SC}$ with its signature pk_{DOS} to AC for consistency verification.

$$proof_{SC} = \{AVF_{c\times t}, h_{\text{root}}^{(ts_1)}, \{h_{\lambda_\varphi}^{(ts_1)}\}_{\varphi\in[1,c]}\}$$

- Upon receiving the proof from SC and TC. AC calculates the verification information $\Omega^{(ts_1)}$ based on the $proof_{SC}$:

$$\Omega^{(ts_1)} = \prod_{\varphi\in[1,c]} g^{\Lambda_\varphi^{(ts_1)} \cdot h_{\lambda_\varphi}^{(ts_1)}} modN.$$

- AC checks the signature of pk_{DO_S} and pk_{DO_T}, then verifies the correctness of $proof_{TC}$ with Equation (4). If the equation verification fails, TC may not store the data as required and AC notifies SC about the exceptional situation; otherwise, AC generates an audit log $rod^{(ts_1)} = \left\{ \text{chal}^{(ts_1)}, \Omega^{(ts_1)}, \text{proof}_{TC}, \text{proof}_{AC}, p_{DO_S}, pk_{DO_T} \right\}$ and stores $rod^{(ts_1)}$ on the blockchain for further checking logs.

$$(\zeta^{(ts_1)})^e \cdot g^{(ts_1)} (\text{mod}\, N) \stackrel{?}{=} \Omega^{(ts_1)} \cdot g^{\mu^{(ts_1)}} (\text{mod}\, N). \quad (4)$$

7. Security Analysis

In this section, we conduct a series of theoretical analyses to prove the security of BDCA. We first provide a security proof of the advanced gamma multi-signature scheme (AGMS) and then prove that BDCA can achieve the design goals defined in Section 5.3. The detailed security proofs are shown as follows:

Theorem 1. *The security of AGMS can be assured under the assumption that the computation of discrete logarithms is a computationally infeasible problem.*

Proof. The idea behind the proof lies in the notion that if it were computationally feasible to fabricate the signature of AGMS, it would imply that adversary $\mathcal{A}$ has the ability to circumvent the discrete logarithm (DL) assumption [39]. Otherwise, if the DL assumption holds, then the security of AGMS can be ensured, thereby preventing potential cross-chain tampering attacks and privacy leakage attacks. We assume that adversary $\mathcal{A}$ has the ability to resolve the DL assumption in $\mathcal{G}$, which can forge a signature $(c^*, \mathrm{sig}^*, hv^*)$ to pass the verification. Thus, we can deduce Equation (5):

$$\begin{aligned} g^{sig} &= V_{\text{leader}}^{c} \cdot PK_{\text{leader}}^{-hv} = V_{\text{leader}}^{c} \prod_{i\in[1,\omega]} pk_i^{-hv} \\ g^{\text{sig}} &= V_{\text{leader}}^{c^*} \cdot PK_{\text{leader}}^{-hv^*} = V_{\text{leader}}^{c^*} \prod_{i\in[1,\omega]} pk_i^{*-hv^*}, \end{aligned} \tag{5}$$

where $\{pk_i\}_{i\in[1,\omega]}$ and $\{pk_i^*\}_{i\in[1,\omega]}$ are two public key sets aiming to forge the signature. We can conclude that the signature can be forged successfully only if $V_{\text{leader}}^{c} = V_{\text{leader}}^{c^*}$, $c^{-1}hv = c^{*-1}hv^*$, and $\left\{pk_i^{-hv}\right\}_{i\in[1,\omega]} = \left\{pk_i^{*-hv^*}\right\}_{i\in[1,\omega]}$. Hence, we can deduce Equation (6).

$$\begin{aligned} V_{\text{leader}} &= g^{\text{sigc}\ -1} \prod_{i\in[1,\omega]} pk_i^{c^{-1}hvs}. \\ V_{\text{leader}} &= g^{\text{sig}\ ^*c^{*-1}} \prod_{i\in[1,\omega]} pk_i^{c^{*-1}hv^*} \end{aligned} \tag{6}$$

Based on Equation (6), we can obtain Equation (7) and it can be asserted that $\mathcal{A}$ has the capability to generate all possible private keys with the exception of its own if $c^* = c$ and $e^* = e$ from Equation (7). According to [40], it can be concluded that the likelihood of the adversary $\mathcal{A}$ succeeding in generating two distinct outputs that satisfy the verification criteria is exceedingly small. As a consequence, the robustness of the AGMS scheme can effectively mitigate the potential risks of cross-chain tampering attacks and privacy leakage attacks.

$$g^{\text{sigc}^{-1} - sig^*c^{*-1}} = \prod_{i\in[1,\omega]} pk_i^{c^{*-1}hv^* - c^{-1}hv} \tag{7}$$

□

Theorem 2. *If all entities of the BDCA have duly complied with the prescribed procedures, it can be asserted that the consistency of dynamic cross-chain data can be reliably ensured.*

Proof. The idea behind the proof lies in the notion that if all entities involved in the BDCA abide by the prescribed procedures in an honest and diligent manner, certain associated parameters can be computed with integrity and accuracy. Under such circumstances, AC is able to ensure the auditing consistency between the source chain (SC) and target chain (TC), thereby promoting the reliability and trustworthiness of cross-chain transactions. In the process of data auditing, AC generates verification information $\Omega^{(ts_1)}$ based on the $proof_{SC}$

and checks the validity of $proof_{TC}$ based on Equation (4). The correctness and soundness of Equation (4) can be deduced as follows:

$$\begin{aligned} LHS &= (\Omega^{(ts_1)})^e \cdot g^{(ts_1)} (modN) \\ &= (\prod_{\varphi \in [1,c]} (((g^{h_{\lambda\varphi}^{(ts_1)}} \cdot g^{m_{\lambda\varphi}^{(ts_1)}})^d)^{\Lambda_\varphi^{(ts_1)}})^e) \cdot g^{r^{(ts_1)}} (modN) \\ &= (\prod_{\varphi \in [1,c]} (g^{h_{\lambda\varphi}^{(ts_1)} \cdot \Lambda_\varphi^{(ts_1)}} \cdot g^{m_{\lambda\varphi}^{(ts_1)} \cdot \Lambda_\varphi^{(ts_1)}})) \cdot g^{r^{(ts_1)}} (modN) \\ &= (\prod_{\varphi \in [1,c]} (g^{h_{\lambda\varphi}^{(ts_1)} \cdot \Lambda_\varphi^{(ts_1)}})) \cdot (g^{\sum_{\varphi \in [1,c]} m_{\lambda\varphi}^{(ts_1)} \cdot \Lambda_\varphi^{(ts_1)}} \cdot g^{r^{(ts_1)}}) (modN) \\ &= \Omega^{(ts_1)} \cdot g^{\mu^{(ts_1)}} (modN) \\ &= RHS \end{aligned} \tag{8}$$

It can be ascertained that the correctness of Equation (4) can be demonstrated with Equation (8) and the assurance of dynamic cross-chain data consistency primarily hinges on establishing the accuracy of Equation (4). Hence, the consistency of dynamic cross-chain data auditing can be guaranteed. □

8. Performance Analysis

We have developed a prototype of the BDCA, utilizing Hyperledger Fabric v1.4 on Ubuntu18.04 with a system memory of 32GB. The implementation leverages the PBC library [41] and simulates the entire experimental procedure by executing three distinct smart contracts, namely S_a, S_s and S_t, which have been developed using the Go programming language [42]. Firstly, we evaluate the efficiency of AGMS, compared with commonly used multi-signature algorithms in the blockchain. Then, we evaluate the computation time and communication overhead of BDCA.

8.1. AGMS Execution Overhead Evaluation

Currently, mainstream multi-signature schemes [43] applied to the blockchain can be divided into RSA-based multi-signature schemes [44], Schnorr-based multi-signature schemes (e.g., Cosi [45], and AGMS), and BLS-based multi-signature schemes [46]. The results are illustrated in Figure 4. The BLS-based multi-signature scheme exhibits significantly longer processing times for both signing and verification operations compared to the other two categories of signature schemes, attributed to the highly time-consuming bilinear pairing operation involved in the BLS-based multi-signature. Despite the fact that the RSA-based multi-signature scheme exhibits the lowest execution time during the verification process, its total time overhead is comparable to that of Cosi and AGMS. The signature length of the RSA-based multi-signature scheme is 2048 bit, while the signature length of the Schnorr-based multi-signature is only 448 bit, also indicating the advantage of the Schnorr-based multi-signature scheme. Moreover, we have taken into account the variation of execution time with the number of signers, as illustrated in Figure 5, while the AGMS signature scheme incurs a higher cost than Cosi, it offers a higher level of security than Cosi, particularly in terms of mitigating the risks of rogue-key attacks and k-sum problem attacks [47], making it a justifiable choice in BDCA where security is of utmost importance.

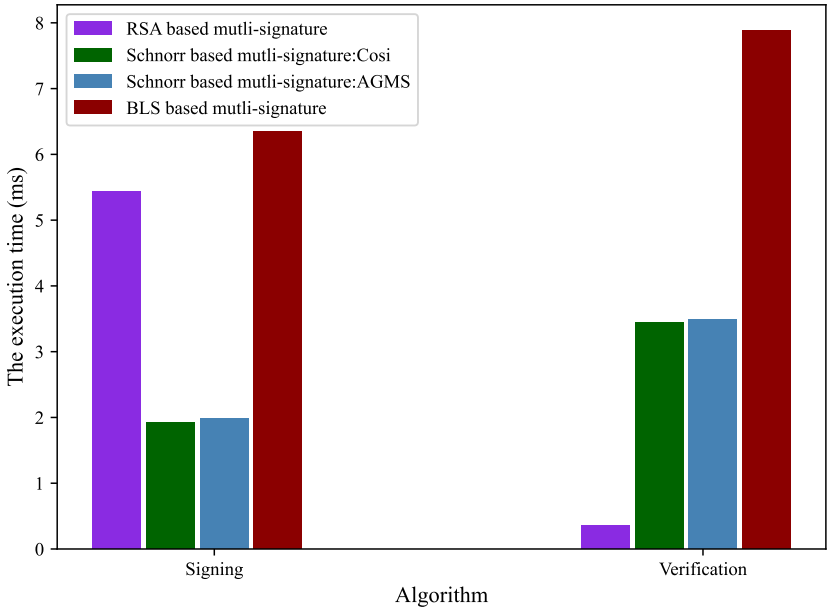


Figure 4. The execution time of typical multi-signature schemes.

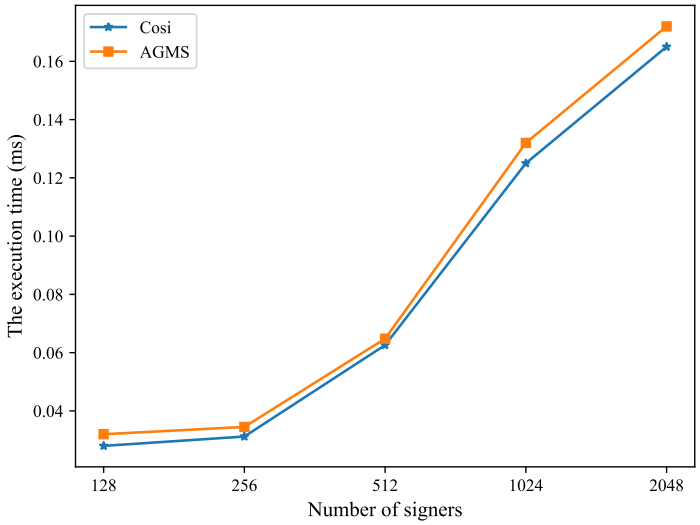


Figure 5. The execution time of Cosi and AGMS in different numbers of signers.

8.2. Computation Overhead Evaluation

We conduct experiments to evaluate the computation overhead of BDCA with a similar scenario as in previous studies using Fortress [48] and CPVPA [38]. As shown in Figure 6, our experimental results indicate that the computation overhead of the BDCA in preprocessing data is lower compared to the Fortress and CPVPA schemes. This is attributed to the fact that the Fortress scheme requires the utilization of zero-knowledge proofs (ZKPs) in preprocessing data while the CPVPA scheme involves multiple costly exponentiation and multiplication operations, resulting in a significantly larger computation overhead. Figure 7 reveals that when the number of challenged blocks is fixed at 300 blocks, irrespective of block size, the computation cost of BDCA in auditing data is marginally superior to that of Fortress. This is due to the fact that the Fortress scheme requires doubling the computations for combined blocks with two different authentication tags, whereas BDCA exhibits a more efficient performance in this regard. Moreover, the computational efficiency of BDCA is significantly higher than that of the CPVPA scheme, which involves multiple computations for combined blocks with different authentication tags. In general, BDCA is efficient in computation.

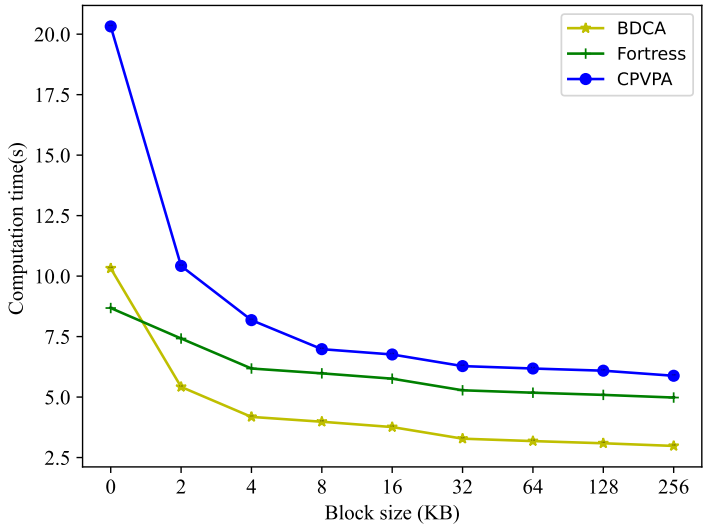


Figure 6. The computation overhead of BDCA in preprocessing data vs. CPVPA and Fortress.

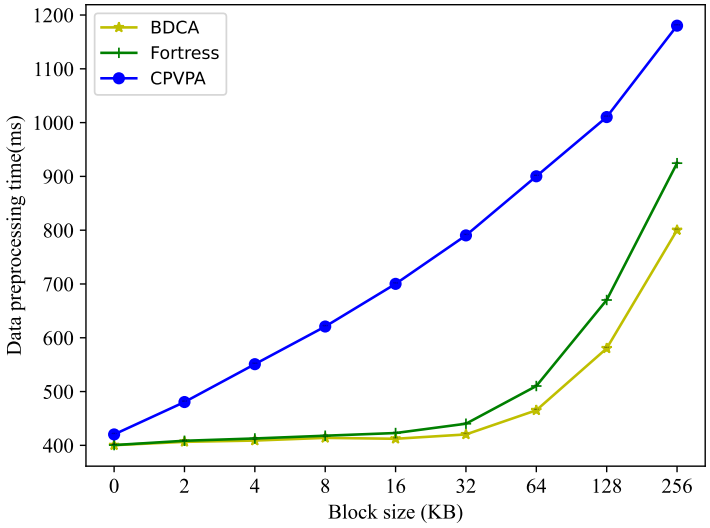


Figure 7. The computation overhead of BDCA in auditing data vs. CPVPA and Fortress.

8.3. Communication Overhead Evaluation

We conduct similar experiments to evaluate the communication overhead of BDCA with Fortress and CPVPA. Figure 8 demonstrates that the communication overhead of the BDCA in preprocessing data is lower compared to the Fortress scheme. This is attributed to the fact that the Fortress scheme requires the generation of various commitments of zero-knowledge proofs (ZKPs), which results in a significant communication burden. In contrast, BDCA only requires transferring the BV-MHT, which incurs a considerably lower communication overhead. Additionally, the communication overhead of the CPVPA scheme grows linearly with the number of blocks, resulting in a higher communication overhead compared to BDCA. Figure 9 highlights that both the Fortress and CPVPA schemes require transferring more combined blocks against each challenge in auditing data, resulting in a higher communication overhead compared to BDCA. This is due to the fact that BDCA employs a more efficient method for generating and transmitting combined blocks. Specifically, BDCA generates and transmits only the necessary combined blocks for each challenge, resulting in a significantly lower communication overhead. In general, BDCA is efficient in communication.

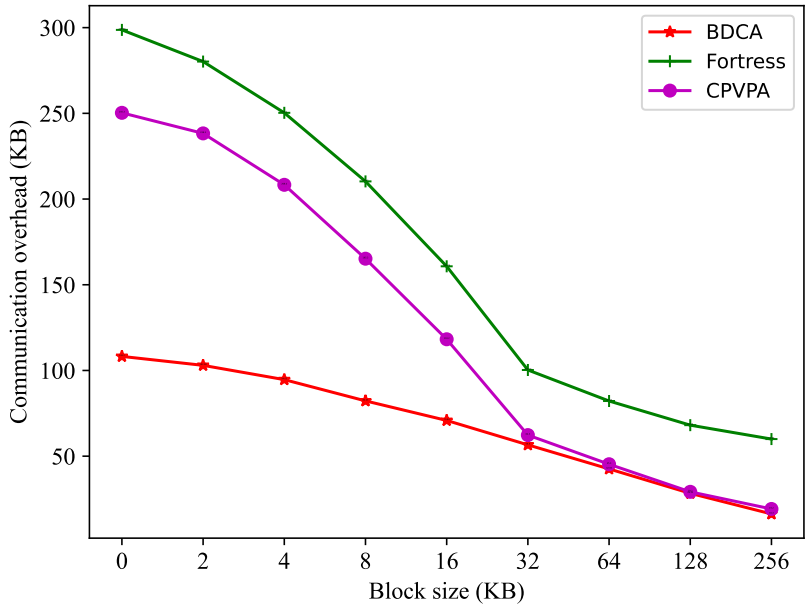


Figure 8. The communication overhead of BDCA in preprocessing data vs. CPVPA and Fortress.

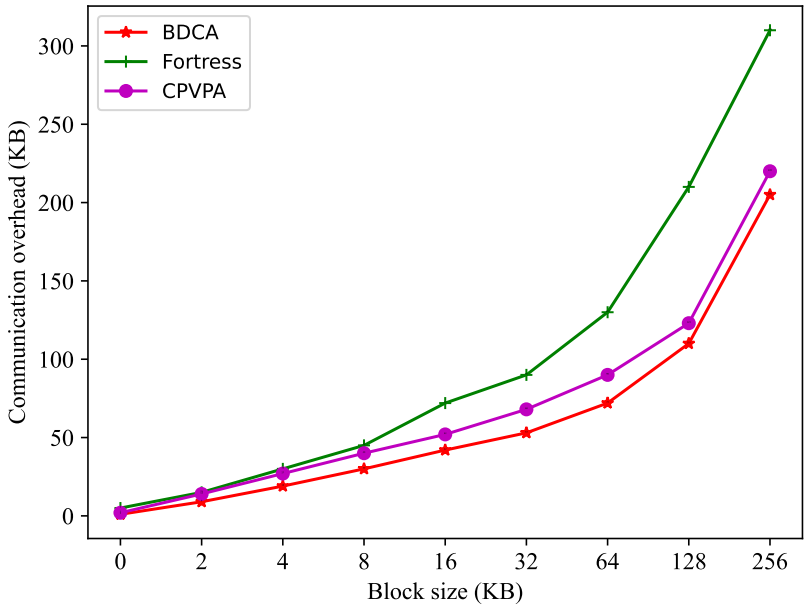


Figure 9. The communication overhead of BDCA in auditing data vs. CPVPA and Fortress.

8.4. The Probability of Data Consistency Guarantees

In Figure 10, our evaluation of the proof generation time under different standards (from the number of challenged blocks equals 240 to 460) of tampered data detection reveals that the number of challenged blocks is the primary determinant of the time required for proof generation. Specifically, increasing the confidence level (from 90 to 99%) [20] of data consistency guarantees leads to a substantial increase in the proof generation time for a fixed data tampering rate of 1%, emphasizing the importance of optimizing consistency guarantees to balance performance and security considerations.

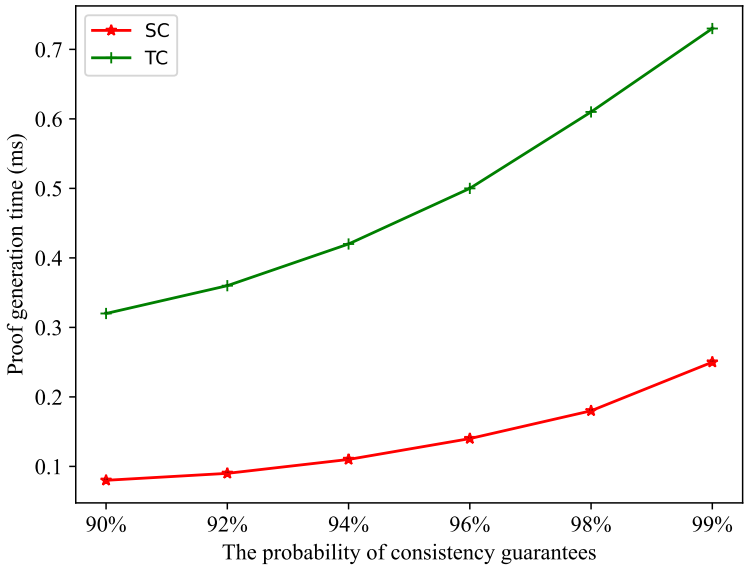


Figure 10. Proof generation time.

9. Conclusions

In this paper, we present a novel blockchain-based distributed model BDCA, that leverages the BV-MHT and AGMS to implement a lightweight and secure data consistency verification process. By incorporating the auditing capabilities of *AC* and the privacy-preserving features of AGMS, BDCA provides a robust and efficient means of ensuring the integrity and confidentiality of cross-chain transactions. Additionally, we introduce BV-MHT and AFV to enable data updates and batch verification with minimal overheads. We also conduct a comprehensive security analysis to demonstrate the security and reliability of BDCA in practice. Experimental results reveal that the average execution time of AGMS scheme is 5.3 ms, which exceeds that of RSA and BLS, and rivals that of Cosi, while the security of AGMS outperforms that of Cosi. Furthermore, BDCA is practical and highly efficient for data preprocessing and auditing in terms of computation and communication, outpacing both CPVPA and Fortress. BDCA can also provide data consistency guarantees of up to 99%. Our future work involves deploying BDCA on a public blockchain platform, e.g., Ethereum, to evaluate its scalability and efficiency in real-world applications. The potential decline in interoperability, scalability, and security of BDCA with the increasing user nodes underscores the need for ongoing research and optimization of BDCA. Some new approaches to improve these metrics will be explored in order to ensure that BDCA can scale effectively and maintain high levels of security and performance.

Author Contributions: Conceptualization, J.Z., Y.Z. and J.J.; methodology, J.Z., Y.Z. and J.J.; validation, J.Z.; formal analysis, J.Z.; investigation, J.Z.; resources, J.Z.; data curation, J.Z.; writing—original draft preparation, J.Z.; writing—review and editing, Y.Z. and J.J.; visualization, J.Z.; supervision, Y.Z.; All authors have read and agreed to the published version of the manuscript.

Funding: This research is funded by the Postgraduate Research & Practice Innovation Program of NUAA xcxjh20221616.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: Thanks to reviewers for their valuable time. Thanks to Yushu Zhang for their invaluable contribution to improving our written English, which enhanced the quality of our manuscript.

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

References

1. Arasteh , H.; Hosseinnezhad, V.; Loia, V.; Tommasetti, A.; Troisi, O.; Shafie-khah, M.; Siano, P. Iot-based smart cities: A survey. In Proceedings of the IEEE 16th International Conference on Environment and Electrical Engineering (EEEIC), Florence, Italy, 7–10 June 2016; pp. 1–6.
2. Devi, Y.U.; Rukmini, M. IoT in connected vehicles: Challenges and issues—A review. In Proceedings of the International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES), Paralakhemundi, India, 3–5 October 2016; pp. 1864–1867.
3. Gandhi, D.A.; Ghosal, M. Intelligent healthcare using IoT: A extensive survey. In Proceedings of the 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), Coimbatore, India, 20–21 April 2018; pp. 800–802.
4. Taherdoost, H. Blockchain-Based Internet of Medical Things. *Appl. Sci.* **2023**, *13*, 1287. [CrossRef]
5. Sadeq, N.; Hamzeh, Z.; Nassreddine, G.; ElHassan, T. The impact of Blockchain technique on trustworthy healthcare sector. *Mesopotamian J. CyberSecurity* **2023**, *2023*, 105–115.
6. Qiao, R.; Zhu, S.; Wang, Q.; Qin, J. Optimization of dynamic data traceability mechanism in Internet of Things based on consortium blockchain. *Int. J. Distrib. Sens. Netw.* **2018**, *14*, 1550147718819072. [CrossRef]
7. Cheng, J.; Li, Y.; Yuan, Y.; Zhang, B.; Xu, X. A Blockchain-Based Trust Model for Uploading Illegal Data Identification. *Appl. Sci.* **2022**, *12*, 9657. [CrossRef]
8. Shafagh, H.; Burkhalter, L.; Hithnawi, A.; Duquennoy, S. Towards blockchain-based auditable storage and sharing of IoT data. In Proceedings of the 2017 on Cloud Computing Security Workshop, Dallas, TX, USA, 3 November 2017; pp. 45–50.
9. Wang, C.; Bi, Z.; Da Xu, L. IoT and cloud computing in automation of assembly modeling systems. *IEEE Trans. Ind. Inform.* **2014**, *10*, 1426–1434. [CrossRef]
10. Aazam, M.; Khan, I.; Alsaffar, A.A.; Huh, E.N. Cloud of Things: Integrating Internet of Things and cloud computing and the issues involved. In Proceedings of the 2014 11th International Bhurban Conference on Applied Sciences & Technology (IBCAST), Islamabad, Pakistan, 14–18 January 2014; pp. 414–419.
11. Mekala, M.S.; Viswanathan, P. A Survey: Smart agriculture IoT with cloud computing. In Proceedings of the 2017 International Conference on Microelectronic Devices, Circuits and Systems (ICMDCS), Vellore, India, 10–12 August 2017; pp. 1–7.
12. AlShamsi, M.; Al-Emran, M.; Shaalan, K. A systematic review on blockchain adoption. *Appl. Sci.* **2022**, *12*, 4245. [CrossRef]
13. Johar, S.; Ahmad, N.; Asher, W.; Cruickshank, H.; Durrani, A. Research and applied perspective to blockchain technology: A comprehensive survey. *Appl. Sci.* **2021**, *11*, 6252. [CrossRef]
14. Chen, R.; Li, Y.; Yu, Y.; Li, H.; Chen, X.; Susilo, W. Blockchain-based dynamic provable data possession for smart cities. *IEEE Internet Things J.* **2020**, *7*, 4143–4154. [CrossRef]
15. Hashim, A.N. Blockchain technology, methodology behind it, and its most extensively used encryption techniques. *Al-Salam J. Eng. Technol.* **2023**, *2*, 140–151.
16. Vaigandla, K.K.; Karne, R.; Siluveru, M.; Kesoju, M. Review on Blockchain Technology: Architecture, Characteristics, Benefits, Algorithms, Challenges and Applications. *Mesopotamian J. CyberSecurity* **2023**, *2023*, 73–85.
17. Hope-Bailie, A.; Thomas, S. Interledger: Creating a standard for payments. In Proceedings of the 25th International Conference Companion on World Wide Web, Montreal, QC, Canada, 11–15 May 2016; pp. 281–282.
18. Back, A.; Corallo, M.; Dashjr, L.; Friedenbach, M.; Maxwell, G.; Miller, A.; Poelstra, A.; Timón, J.; Wuille, P. Enabling Blockchain Innovations with Pegged Sidechains. 2014. Available online: http://www.cpensciencereview.com/papers/123/enablingblockchain-innov (accessed on 18 May 2023).
19. Poon, J.; Dryja, T. The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments. 2016. Available online: https://www.bitcoinlightning.com (accessed on 18 May 2023).
20. Ateniese, G.; Burns, R.; Curtmola, R.; Herring, J.; Kissner, L.; Peterson, Z.; Song, D. Provable data possession at untrusted stores. In Proceedings of the 14th ACM Conference on Computer and Communications Security, Alexandria, VA, USA, 31 October–2 November 2007; pp. 598–609.
21. Jiang, Y.; Wang, C.; Wang, Y.; Gao, L. A cross-chain solution to integrating multiple blockchains for IoT data management. *Sensors* **2019**, *19*, 2042. [CrossRef]
22. Tian, H.; Xue, K.; Luo, X.; Li, S.; Xu, J.; Liu, J.; Zhao, J.; Wei, D.S. Enabling cross-chain transactions: A decentralized cryptocurrency exchange protocol. *IEEE Trans. Inf. Forensics Secur.* **2021**, *16*, 3928–3941. [CrossRef]
23. Xiong, A.; Liu, G.; Zhu, Q.; Jing, A.; Loke, S.W. A notary group-based cross-chain mechanism. *Dig. Commun. Netw.* **2022**, *8*, 1059–1067. [CrossRef]
24. Herlihy, M.; Liskov, B.; Shrira, L. Cross-chain deals and adversarial commerce. *arXiv* **2019**, arXiv:1905.09743.
25. Li, Y.; Weng, J.; Li, M.; Wu, W.; Weng, J.; Liu, J.N.; Hu, S. ZeroCross: A sidechain-based privacy-preserving Cross-chain solution for Monero. *J. Parallel Distrib. Comput.* **2022**, *169*, 301–316. [CrossRef]
26. Ateniese, G.; Di Pietro, R.; Mancini, L.V.; Tsudik, G. Scalable and efficient provable data possession. In Proceedings of the 4th International Conference on Security and Privacy in Communication Netowrks, Istanbul, Turkey, 22–25 September 2008; pp. 1–10.
27. Tian, H.; Chen, Y.; Chang, C.C.; Jiang, H.; Huang, Y.; Chen, Y.; Liu, J. Dynamic-hash-table based public auditing for secure cloud storage. *IEEE Trans. Serv. Comput.* **2015**, *10*, 701–714. [CrossRef]

28. Rao, L.; Zhang, H.; Tu, T. Dynamic outsourced auditing services for cloud storage based on batch-leaves-authenticated Merkle hash tree. *IEEE Trans. Serv. Comput.* **2017**, *13*, 451–463. [CrossRef]
29. Shen, J.; Shen, J.; Chen, X.; Huang, X.; Susilo, W. An efficient public auditing protocol with novel dynamic structure for cloud data. *IEEE Trans. Inf. Forensics Secur.* **2017**, *12*, 2402–2415. [CrossRef]
30. Yao, A.C.C.; Zhao, Y. Online/offline signatures for low-power devices. *IEEE Trans. Inf. Forensics Secur.* **2012**, *8*, 283–294. [CrossRef]
31. Schnorr, C.P. Efficient signature generation by smart cards. *J. Cryptol.* **1991**, *4*, 161–174. [CrossRef]
32. Merkle, R.C. A certified digital signature. In *Proceedings of the Conference on the Theory and Application of Cryptology*; Springer: Berlin/Heidelberg, Germany, 1989; pp. 218–238.
33. Li, H.; Lu, R.; Zhou, L.; Yang, B.; Shen, X. An efficient merkle-tree-based authentication scheme for smart grid. *IEEE Syst. J.* **2013**, *8*, 655–663. [CrossRef]
34. Xiao, Y.; Zhang, P.; Liu, Y. Secure and efficient multi-signature schemes for fabric: An enterprise blockchain platform. *IEEE Trans. Inf. Forensics Secur.* **2020**, *16*, 1782–1794. [CrossRef]
35. Schnorr, C.P. Efficient identification and signatures for smart cards. In *Proceedings of the Advances in Cryptology—CRYPTO'89*; Proceedings 9; Springer: Berlin/Heidelberg, Germany, 1990; pp. 239–252.
36. Wang, W.; Zhang, Z.; Wang, G.; Yuan, Y. Efficient cross-chain transaction processing on blockchains. *Appl. Sci.* **2022**, *12*, 4434. [CrossRef]
37. Erway, C.C.; Küpçü, A.; Papamanthou, C.; Tamassia, R. Dynamic provable data possession. *ACM Trans. Inf. Syst. Secur. (TISSEC)* **2015**, *17*, 1–29. [CrossRef]
38. Zhang, Y.; Xu, C.; Lin, X.; Shen, X. Blockchain-based public integrity verification for cloud storage against procrastinating auditors. *IEEE Trans. Cloud Comput.* **2019**, *9*, 923–937. [CrossRef]
39. Diffie, W.; Hellman, M.E. New directions in cryptography. In *Democratizing Cryptography: The Work of Whitfield Diffie and Martin Hellman*; Association for Computing Machinery: New York, NY, USA, 2022; pp. 365–390.
40. Bellare, M.; Neven, G. Multi-signatures in the plain public-key model and a general forking lemma. In Proceedings of the 13th ACM conference on Computer and Communications Security, Alexandria, VA, USA, 30 October–3 November 2006; pp. 390–399.
41. Lynn, B. The Pairing-Based Cryptography (PBC) Library. 2013. Available online: http://crypto.stanford.edu/pbc (accessed on 18 May 2023).
42. Xiong, H.; Jin, C.; Alazab, M.; Yeh, K.H.; Wang, H.; Gadekallu, T.R.; Wang, W.; Su, C. On the design of blockchain-based ECDSA with fault-tolerant batch verification protocol for blockchain-enabled IoMT. *IEEE J. Biomed. Health Inform.* **2021**, *26*, 1977–1986. [CrossRef]
43. Abdul-Sada, H.H.; Rabee, F. The Genetic Algorithm Implementation in Smart Contract for the Blockchain Technology. *Al-Salam J. Eng. Technol.* **2023**, *2*, 37–47.
44. Hohenberger, S.; Waters, B. Synchronized aggregate signatures from the RSA assumption. In Proceedings of the Advances in Cryptology–EUROCRYPT 2018: 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, 29 April–3 May 2018; Proceedings, Part II; Springer: Berlin/Heidelberg, Germany, 2018; pp. 197–229.
45. Syta, E.; Tamas, I.; Visher, D.; Wolinsky, D.I.; Jovanovic, P.; Gasser, L.; Gailly, N.; Khoffi, I.; Ford, B. Keeping authorities "honest or bust" with decentralized witness cosigning. In Proceedings of the 2016 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–26 May 2016; pp. 526–545.
46. Boneh, D.; Drijvers, M.; Neven, G. Compact multi-signatures for smaller blockchains. In Proceedings of the Advances in Cryptology–ASIACRYPT 2018: 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, 2–6 December 2018; pp. 435–464.
47. Drijvers, M.; Edalatnejad, K.; Ford, B.; Kiltz, E.; Loss, J.; Neven, G.; Stepanovs, I. On the security of two-round multi-signatures. In Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 20–22 May 2019; pp. 1084–1101.
48. Armknecht, F.; Bohli, J.M.; Karame, G.O.; Liu, Z.; Reuter, C.A. Outsourced proofs of retrievability. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, 3–7 November 2014; pp. 831–843.