BeDCV: Blockchain-Enabled Decentralized Consistency Verification for Cross-Chain Calculation

Yushu Zhang, Jiajia Jiang, Xuewen Dong, Liangmin Wang, and Yong Xiang

Abstract—With the increase of data stored on the blockchain, the efficiency of storage and calculation of blockchain has gradually become a bottleneck restricting the development of blockchain. By storing data on multiple chains, blockchains can request data from other chains for calculation and the storage pressure can be alleviated. But the transfer of a large amount of data between chains suffers from low transfer efficiency and poor security. A reasonable design is to perform the calculation on the data storage chain and only transfer the results across chains. However, since the calculation process is invisible, blockchains cannot judge the consistency of calculation results from other chains. In this paper, we provide a blockchain-enabled decentralized consistency verification scheme for cross-chain calculation (BeDCV). Considering the decentralized characteristic of blockchain, we adopt the blockchain called *supervision chain* for decentralized auditing. We modify paillier homomorphic encryption to encrypt data involved in the calculation for correctness verification. Then, we aggregate the ciphertexts of data to generate the audit proof for integrity verification. Besides, we verify whether the data involved in the calculation are real-time by leveraging a counting bloom filter. The supervision chain can check the correctness, integrity, and real-time performance of cross-chain data calculation without revealing any original information about the data. The theoretical and experimental analysis demonstrates that BeDCV can verify the consistency of cross-chain data calculation result effectively, realizing secure and reliable expansion of blockchain.

Index Terms—Consistency verification, homomorphic encryption, data integrity, data correctness, bloom filter, blockchain-enabled auditing, cross chain

1 Introduction

SINCE blockchain is a cross-network distributed collaboration system that emphasizes security, availability, fault tolerance, consistency, and transactionality, it is equipped with more complex algorithms and tedious multi-participant collaboration to achieve credibility, unforgeability, and traceability. The distributed CAP principle [1] notes that it is impossible to achieve consistency, availability, and partition tolerance simultaneously in a distributed system. Therefore, under the same hardware resource investment, the performance of the blockchain is often lower than that of the centralized system.

Under the single-chain architecture, the operating speed of the blockchain network can reach a relatively high-performance level, meeting the business needs of general scenarios. However, when the number of users and transaction throughput in the blockchain network is large, the storage and calculation performance of blockchain, especially that of a single chain, gradually becomes a bottleneck restricting the further development of blockchain network. Therefore, *multi-chain parallelism* is proposed to solve the limitation, which realizes the expansion of blockchain functions and breaks through the performance bottleneck of the single-chain architecture.

For example, in the field of logistics, small blockchain systems maintained by storage companies, freight companies, and others transfer data stored on the chain to the main chain of the logistics trading platform. However, sending a large amount of data to the main chain for data calculation will increase the communication cost of cross-chain interaction and the calculation burden of the main chain. Therefore, we choose to hand over part of the calculation work to the side chain that stores the data and directly transfers the data calculation results during the cross-chain interaction. However, since the process of data calculation is agnostic to the main chain, it cannot judge the reliability of the received data calculation results. Accordingly, how to effectively verify the consistency of the result of cross-chain data calculation is heavily demanded.

Currently, the mainstream cross-chain technologies include notary schemes [2], sidechains/relays [3], and hash-locking [4]. By connecting independent blockchain networks, cross chain realizes the interaction between blockchains, alleviating various problems caused by performance bottlenecks in a single blockchain network. However, most applications of these cross-chain technologies focus on asset transfer rather than information calls. In existing multi-chain schemes, some studies utilized smart contracts

- *Y. Zhang and J. Jiang are with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, 210023, China. E-mail: yushu@nuaa.edu.cn; Jiangjiajia@nuaa.edu.cn.*
- *X. Dong is with the School of Computer Science and Technology, Xidian University, the Shaanxi Key Laboratory of Network and System Security, Xi'an, 710071, China. E-mail: xwdong@xidian.edu.cn.*
- *L. Wang is with the School of Cyber Science and Engineering, Southeast University, Nanjing, 211189, China. E-mail: wanglm.seu@foxmail.com.*
- *Y. Xiang is with the School of Information Technology, Deakin University, Melbourne, VIC 3125, Australia. E-mail: yong.xiang@deakin.edu.au.*

This research was supported by the National Key R&D Program of China (No. 2020YFB1005500).

to complete the basic security guarantee for cross-chain interaction but did not give a specific implementation method to ensure the consistency of cross-chain information [5], [6].

The main application scenario of data consistency verification is to verify the integrity of data stored in the cloud. Previous studies usually introduced third-party auditors to verify the integrity of the data without obtaining complete data information [7], [8], [9]. The traditional verification schemes audit the integrity of the data stored by the data receiver. On the contrary, we need to audit the behavior of the chain that transfers data in cross-chain interaction. As a result, it cannot be directly applied to multi-chain scenarios.

Besides, the use of third-party auditors in cross-chain scenarios weakens the decentralization of the blockchain system, and there is also the possibility that the auditors are malicious. Many researchers have utilized blockchain for auditing [10], [11]. With the help of smart contracts and consensus mechanisms on blockchains, the laws and contracts for auditing blockchains are transformed into simple and deterministic code-based rules, which are automatically executed by the underlying blockchain network, ensuring that the blockchain always outputs a fair audit result, which ensures the decentralization of the entire scheme in cross-chain interaction.

In this paper, we propose a blockchain-enabled decentralized consistency verification scheme (BeDCV) by adopting the blockchain called *supervision chain* to audit the process of cross-chain data calculation. BeDCV focuses on realizing the correctness, integrity, and real-time audit of cross-chain data calculation. We apply paillier homomorphic encryption to encrypt the data to be calculated and aggregate the ciphertexts to generate the integrity audit proof. Additionally, we keep the real-time data information in a set stored in the block header of the latest block with a counting bloom filter. All audit and verification work is done by the supervision chain, which honestly returns unbiased audit results to the participants of the cross-chain interaction. In summary, we make the following contributions:

(1) We modify paillier homomorphic encryption to achieve the correctness verification of cross-chain calculation. The supervision chain can verify the correctness of the calculation result without revealing any information about the original data.
(2) We aggregate the encrypted data to generate the audit proof for integrity verification. The supervision chain uses bilinear pairing to verify the integrity of the aggregated data, ensuring that the audit data sent to the supervision chain are complete.
(3) We design a real-time verification method of data stored on the blockchain with a counting bloom filter to ensure that the data used for calculation and cross-chain are up-to-date.
(4) We propose a blockchain-enabled decentralized audit scheme for cross-chain calculation (BeDCV), which enables accurate and reliable audit of cross-chain calculation.

The paper has been organized in the following way. Section 2 reviews the related work and research status of this paper. Section 3 is concerned with the technologies used in BeDCV. Section 4 defines the system model, threat model, and design goals of BeDCV. Section 5 presents the detailed construction of BeDCV. Section 6 conducts a detailed security analysis in terms of privacy, correctness, integrity, and real-time performance of cross-chain calculation. Section 7 evaluates the performance of BeDCV. Section 8 concludes this paper and points out the future research directions.

2 Related Work

In this part, we summarize the existing blockchain cross-chain technologies and applications and enumerate technologies such as homomorphic encryption and blockchain-based data integrity verification used in BeDCV.

2.1 Cross-chain Technologies and Applications

With the increasing number of blockchains, digital asset transfer and cross-chain communication are inevitably required between chains [12]. To solve the bottleneck of blockchain performance and break the gap between different blockchains, various cross-chain and multi-chain technologies have emerged, which have promoted the expansion of blockchain performance.

In 2016, Tsai et al. [13] proposed a new scheme Beihang Chain. Through the Sharding strategy based on Account-Blockchain and TradingBlockchain, the need of scalability is realized. However, Beihang Chain is mainly aimed at financial business and does not involve legal verification support for complex transactions of diversified digital assets. In 2017, Polkadot[1] and Cosmos[2] were proposed to build basic platforms for cross-chain interaction. Polkadot is designed to enable arbitrary message passing between parachains. Cosmos, on the other hand, focuses on cross-chain asset transfer, and its protocol is relatively simple.

The technologies above achieve cross-chain communication but do not consider the consistency of data in cross-chain interaction. The methods mentioned above solve the limitation of data silos between chains and enable information exchange between chains in a secure and trustless manner, but they do not provide a security mechanism to ensure the consistency of data transmission between chains.

2.2 Homomorphic Encryption

In the audit of cross-chain interaction, if we use the original data for consistency verification, there may be security risks of privacy leakage. In homomorphic encryption, the calculation result of the original data is equal to the decryption result after the calculation on ciphertexts. This feature allows untrusted third parties to directly operate on the ciphertexts without the private key, avoiding the disclosure of sensitive information.

Many well-known public-key encryption technologies are homomorphic, such as RSA [14] and EIGamal [15]. However, they are partial homomorphic encryption, which means they only support addition or multiplication on the ciphertext. In scenarios where complex calculations are not required, partial homomorphic encryption can achieve efficient data calculation [16], [17], [18], [19]. In 2009, Craig

1. https://polkadot.network/
2. https://cosmos.network/

Gentry proposed a feasible fully homomorphic encryption (FHE) scheme for the first time [20], which realizes unlimited ciphertexts addition and multiplication at the same time. Since then, the fully homomorphic encryption scheme has been making great progress for more than ten years [21], [22]. So far, a variety of feasible FHE technologies have been realized. However, since managing the continuously growing noise in ciphertext computing requires a lot of computing resources [23], the efficiency of FHE is still low and cannot meet the needs of practical applications.

2.3 Blockchain-based Data Integrity Verification

With the emergence of blockchain, its decentralization and immutable characteristics provide new solutions for data verification. The decentralized character allows data audits to be dependent on third-party auditors and enhances the security of data. In 2017, Sutton and Samvi described a blockchain-based verification method to audit trails [24]. They pay special attention to privacy, emphasizing the non-repudiation of privacy audit logs. In the same year, Liang *et al.* proposed a cloud environment security framework ProveChain, which audits the integrity of data by blockchain to improve the security of the cloud storage environment [25]. In 2019, Zhang *et al.* proposed a certificateless public verification scheme against procrastinating auditors by the use of blockchain, which enables users to check whether auditors perform the verifications at the prescribed time [10]. In 2020, Wang *et al.* designed a blockchain-based private PDP scheme by leveraging RSA and quadratic residue group modulo the big composite number N [8]. In 2021, Du *et al.* proposed a zero-knowledge storage auditing scheme with an on-chain check mechanism to address the security problem of storage freeriding in the settings of rational actors [11]. However, these blockchain-based audits mostly consider the consistency of data in cloud storage rather than the consistency of data during cross-chain interaction.

3 Priliminaries

3.1 Bilinear Pairing

Bilinear pairing [26] is based on cryptography, which relies on a difficult hypothesis similar to the elliptic curve discrete logarithm problem, which can often be used to reduce the problem in one group to an easier problem in another group. Let $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$, where $\mathbb{G}$ and $\mathbb{G}_1$ are two cyclic addictive groups generated with the same prime order p. A bilinear map has the following properties:

(1) Bilinearity: For all $g_1, g_2 \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$.
(2) Non-degeneracy: For $g \in \mathbb{G}$, $e(g, g) \neq 1$, where 1 is the identity element of group $\mathbb{G}_1$.
(3) Computability: There exists an efficient algorithm that can compute e.

3.2 Paillier Homomorphic Cryptosystem

Paillier homomorphic cryptosystem [16] is based on the quadratic residuosity problem, in which the product of two ciphertexts decrypt to the sum of the corresponding plaintexts. The details of the paillier cryptosystem are described as follows:

• **Key Generation:** Select two large prime numbers p and q randomly, such that $gcd(pq, (p-1)(q-1)) = 1$, compute $N = pq$ and $\lambda = lcm(p-1, q-1)$, where lcm is the least common multiple. Select a random integer $g \in \mathbb{Z}^*_{N^2}$, where N is divided by the order of g, through checking the existence of the following modular multiplicative inverse:

$$\mu = (L(g^{\lambda}(mod\ N^2)))^{-1}(mod\ N^2),$$

where function L is defined as

$$L(u) = (u-1)/N.$$

Thus, the public key is (N, g) and the private key is (λ, μ).

• **Encryption:** For any message $m \in \mathbb{Z}_N$, randomly select $r \in \mathbb{Z}^*_{N^2}$ and compute the ciphertext of m as follows:

$$C = g^m \cdot r^N mod\ N^2.$$

• **Decryption:** Let C be the ciphertext to be decrypted, where $C \in \mathbb{Z}^*_{N^2}$, compute the plaintext as follows:

$$m = L(C^{\lambda}(mod\ N^2)) \cdot \mu\ mod\ N.$$

• **Homomorphic property:** Paillier cryptosystem inherently owns additive homomorphic properties. Particularly, the product of two ciphertexts will be decrypted to the sum of the corresponding plaintexts, i.e., $D(E(m_1, pk) \cdot E(m_2, pk)(mod\ N^2)) = m_1 + m_2(mod\ N)$, because

$$\begin{aligned} E(m_1, pk) \cdot E(m_2, pk) &= (g^{m_1} r_1^N)(g^{m_2} r_2^N)(mod\ N^2) \\ &= g^{m_1+m_2}(r_1 r_2)^N (mod\ N^2) \\ &= E(m_1 + m_2, pk) \end{aligned}$$

Fully homomorphic encryption (FHE) can realize unlimited ciphertexts addition and multiplication at the same time [20], [27]. However, the efficiency of FHE is low and cannot meet the needs of practical applications. The proposed scheme mainly concentrates on the consistency of the calculation result in the cross-chain process. Therefore, to improve the experimental efficiency, we adopt paillier homomorphic encryption in our scheme, which has achieved considerable encryption efficiency and has been practically applied to real life [16]. In theory, any form of homomorphic encryption is suitable for this system.

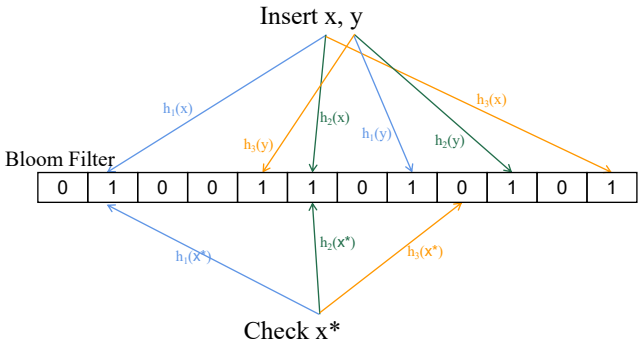

Fig. 1. The technical principle of bloom filter

3.3 Bloom filter

Bloom Filter [28] is a binary-based data structure with good spatial and temporal efficiency. It is usually used to judge the attribution of an element to a set. The technical details of the bloom filter are illustrated in Fig. 1.

A bloom filter uses an m-bit array to represent a set $S = \{m_1, m_2, \ldots, m_n\}$ of n elements. Initially, all the bits in the filter are set to zero, and then we select n hash functions $h_i(x)$, where $i \in [1, n]$, to map items $x \in S$ to random number uniform in range $1, 2, \ldots, m$. An element $x \in S$ is inserted into the filter by setting the bits $h_i(x)$ to one for $i \in [1, n]$. When judging whether the element $y \in S$, we need to check each bit $h_i(y)$ in the set. If any one of them is zero, it means that the element y is not in this set.

4 MODELS

4.1 System Model

In this paper, we consider the consistency audit method of the cross-chain data calculation results in multi-chain scenarios. We apply Hyperledger Fabric to build three consortium blockchains: a main chain (C_{main}), a side chain (C_{side}), and a supervision chain (SC). Each blockchain deploys a smart contract for consistent policing. As shown in Fig. 2, off-chain users participate in the scheme for distributing tasks and performing calculations off-chain.

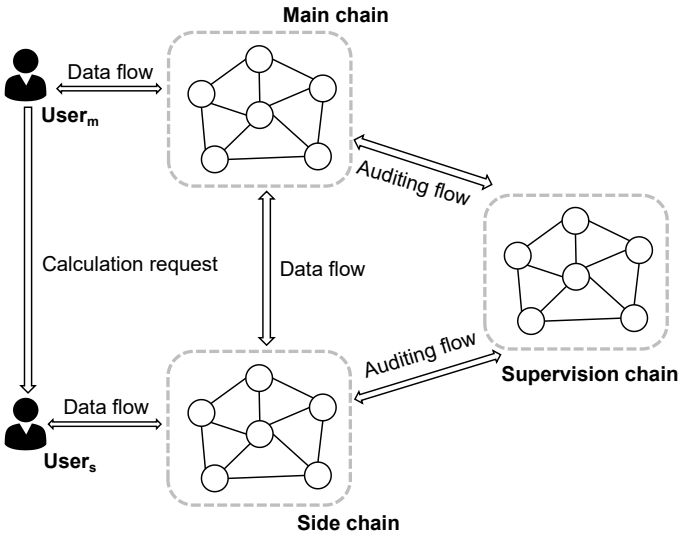

Fig. 2. System model

Main chain (C_{main}): It is a cross-chain data calculation result receiving platform. It provides the timestamp of the latest block for users to generate a calculation task to C_{side} and SC and receives the calculation result from C_{side}.

Side chain (C_{side}): It is an independent blockchain that receives task requests from C_{main}. Users on C_{side} complete the data calculation task and perform the paillier homomorphic encryption off-chain. The result and ciphertexts of paillier are stored on C_{side}. C_{side} is responsible for generating audit proof and sending information to C_{main} and SC.

Supervision chain (SC): It is an auditing platform for the consistency of the cross-chain data calculation result, which is jointly established by national regulatory authorities and centralized institutions in various fields. It receives the audit request R from C_{main} and C_{main}. After the cross-chain interaction is completed, SC receives the audit proof from C_{side} and completes the consistency verification to produce a reasonable judgment of this cross-chain calculation. When there is no audit request, SC operates independently and does not participate in the work of C_{main} and C_{side}.

Off-chain users: Users of C_{main} and C_{side} are responsible for calculation tasks distribution and data calculation. $User_m$ of C_{main} obtains the timestamp of the latest block from C_{main}, and sends the timestamp and calculation task to $User_s$ of C_{side}. $User_s$ obtains data for calculation from C_{side} and completes the calculation and homomorphic encryption off-chain. After the calculation, $User_s$ stores the result into C_{side}.

4.2 Threat Model

In BeDCV, SC is responsible to audit the cross-chain interaction between C_{main} and C_{side} to fill the audit gap in the development of the blockchain. Since SC is established by national regulatory authorities and centralized institutions in various fields with the properties of immutability and transparency, it is considered to be reliable and honest. Generally, there are three forms of cross-chain attacks:

(1) The calculation result of C_{side} is incorrect. Users may not perform data calculations as required to save computing resources, thus providing an incorrect calculation result to C_{main}.
(2) The audit proof provided by C_{side} is incomplete. To pass the correctness verification, C_{side} may provide incomplete data to forge the audit proof.
(3) The data used by C_{side} for calculation are not real-time. C_{side} may perform the calculation with outdated data or directly transmit an outdated data calculation result so that the result received by C_{main} is incorrect.

4.3 Design Goals

Based on the analysis of the proposed cross-chain interaction model, we propose the following goals to ensure data integrity in cross-chain interaction:

- **Correctness.** SC can verify the correctness of the calculation result provided by C_{side} without knowing the original data and the calculation result.
- **Integrity.** SC can verify the integrity of the data involved in the calculation without obtaining the complete data, ensuring that the result of correctness verification is correct.
- **Real-time.** Blockchain records the real-time data within a limited block space. When cross-chain interaction happens, the real-time performance of data can be verified by carrying a small amount of validation information.

5 THE PROPOSED SCHEME

In this section, we give a detailed description of BeDCV. To start with, we give an overview of the consistency verification framework of cross-chain calculation. Then, we analyze

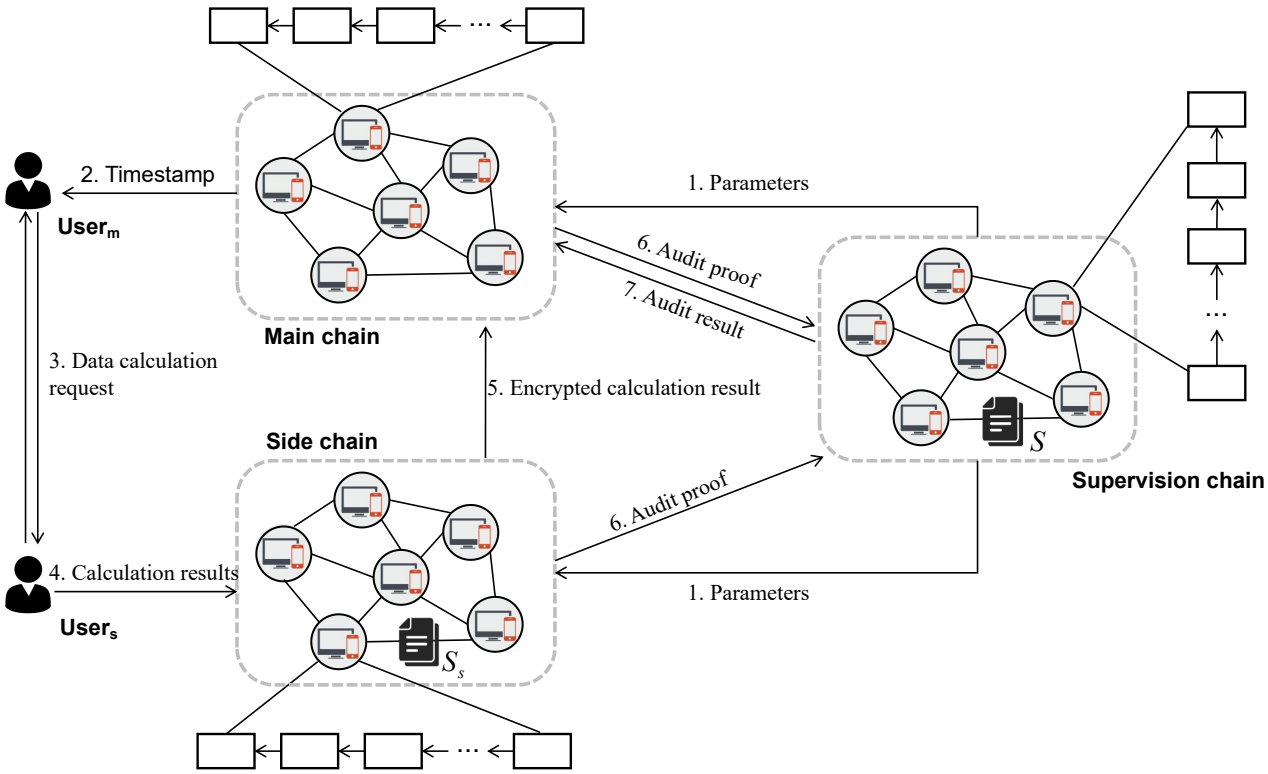

Fig. 3. Overview of the proposed scheme

the consistency of cross-chain calculation from three perspectives: correctness, integrity, and real-time performance. Finally, we elaborate on the whole process of BeDCV in detail.

5.1 Overview

First of all, we provide an overview of BeDCV, which consists of six algorithms: **Initialization**, **Task Requesting**, **Data Calculation**, **Data Writing**, **Results Receiving**, and **Verification**. The overview of BeDCV is illustrated in Fig. 3.

In **Initialization**, SC is responsible for generating a series of security parameters for consistency auditing. Then, it sends the generated parameters to C_{side} and SC.

In **Task Requesting**, $user_m$ obtains the timestamp T from the latest block of C_{main} and packs the calculation task $Task$ and timestamp T into R. Then, $user_m$ generates the private key (λ, μ) and public key N of paillier encryption. Finally, $user_m$ sends the calculation request R and the public key N to SC and $user_s$ of C_{side}. The timestamp T is used for generating time-dependent proof, ensuring that the calculation result sent from C_{side} is generated up-to-date.

In **Data Calculation**, when $user_s$ receives request R and the public key N, it first gets the data $\{m_i\}_{i=1}^{k}$ from C_{side} and completes the specified calculation according to $Task$ to obtain the calculation result M. Then, it performs paillier homomorphic encryption on data $\{m_i\}_{i=1}^{k}$ involved in the calculation to obtain the ciphertexts $\{c_i\}_{i=1}^{k}$ and generates authenticators of $\{c_i\}_{i=1}^{k}$ as $\{\sigma_i\}_{i=1}^{k}$. The data involved in the calculation are mapped into a new counting bloom filter B_s. $User_s$ sends M, $\{c_i\}_{i=1}^{k}$, $\{\sigma_i\}_{i=1}^{k}$, $\{name_i\}_{i=1}^{k}$, $\{\beta_i\}_{i=1}^{k}$, N, h, υ, and B_s to C_{side}.

In **Data Writing**, since the database used in Hyperledger Fabric is Level DB or Couch DB, both of which are non-relational databases based on key-value pairs, C_{side} stores information from $user_s$ in the format of (key = T, value = $(M, \{c_i\}_{i=1}^{k}, \{\sigma_i\}_{i=1}^{k}, \{name_i\}_{i=1}^{k}, \{\beta_i\}_{i=1}^{k}, N, h, \upsilon, B_s)$). C_{side} obtains the corresponding value according to the timestamp T and calculates the audit proof with the determined audit proof generation algorithm by invoking smart contract S_s. Besides, considering the risks of privacy leakage and tampering during cross-chain interaction, smart contract S_s encrypts M with paillier encryption to C and sends C to C_{main}. Finally, C_{side} sends the audit proof to SC.

In **Results Receiving**, $user_m$ of C_{main} can get the ciphertext C from C_{main} and decrypt C to get the plaintext M_r with private keys λ and μ. Before auditing, C_{main} sends the received C to SC as the audit proof.

In **Verification**, the smart contract S on SC automatically performs the consistency verification according to the determined algorithm with the received audit proof from C_{side} and C_{main}. Based on the homomorphic property of paillier, the correctness of the calculation result can be verified without obtaining the original data on C_{side}. The integrity of the data involved in the calculation can be verified by using bilinear pairing with the proof provided by C_{side}. In addition, the counting bloom filter proves that the data involved in this calculation are real-time. The verifications above ensure the consistency of cross-chain calculation.

5.2 Correctness Verification

To verify the correctness of the calculation result provided by C_{side}, we need to provide the audit proof about the

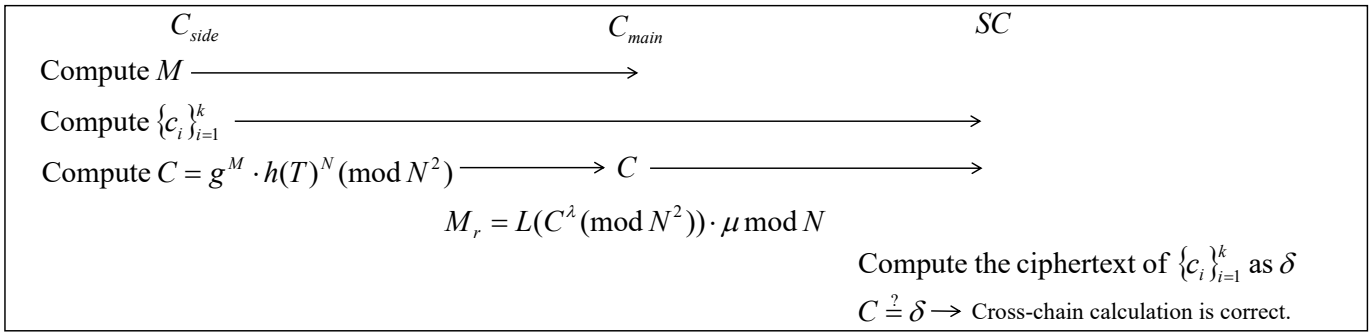

Fig. 4. The Procedure of Correctness Verification

original data involved in the calculation to SC. However, transferring the original data directly may cause information leakage, so we adopt the efficient paillier homomorphic encryption to encrypt the data involved in the calculation. SC can perform the calculation on ciphertexts without obtaining the original data to audit the correctness of the calculation result.

We modify paillier homomorphic encryption by replacing the random r with $h(T)$. $h()$ is a collision-resistant hash function that always generates a random value with the input T. Since r and $h()$ are both randomly generated, the security of the modified paillier can still be guaranteed. It should be noted that paillier is partially homomorphic and supports additive-only ciphertext homomorphic operations, so this scheme currently only supports addition operations.

After the calculation is completed, smart contract S_s is invoked to process the result M into C with paillier homomorphic encryption, in case the calculation result is maliciously tampered with or reveals privacy in the process of cross-chain interaction. Then, C_{side} sends C to C_{main} and sends $\{c_i\}_{i=1}^k$ to SC. When C_{main} receives the data calculation result from C_{side}, it sends C to SC as the audit proof.

Once receiving the audit proof from C_{side} and C_{main}, SC performs the calculation with the ciphertexts $\{c_i\}_{i=1}^k$ to obtain the result δ of ciphertexts. SC compares δ with C from C_{main} to judge whether the cross-chain calculation is correct. The details of correctness verification are shown in Fig. 4.

5.3 Integrity Verification

To accurately verify the correctness of the cross-chain calculation result, C_{side} needs to provide ciphertexts of the original data to SC for homomorphic calculation. However, if the calculation result provided by C_{side} is incorrect, C_{side} may not provide correct or complete audit ciphertexts in an attempt to pass the correctness verification.

We design an integrity verification scheme for the proof provided by C_{side}. C_{side} invokes smart contract S_s to aggregate ciphertexts $\{c_i\}_{i=1}^k$ together with authenticators $\{\sigma_i\}_{i=1}^k$ signed from ciphertexts. C_{side} sends the proof (σ, ς) to SC. When SC receives the audit proof, it uses bilinear pairing to verify the integrity of the data. During this process, any encrypted data will not be maliciously modified or replaced, thus C_{side} could honestly aggregate all ciphertexts and send them to SC instead of replacing them with other information. The details of integrity verification are shown in Figure 5.

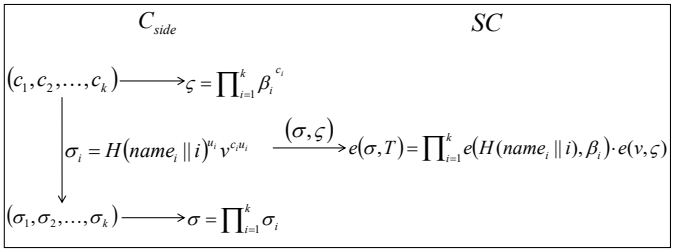

Fig. 5. The Procedure of Integrity Verification

5.4 Real-time Guarantee

If C_{side} uses outdated data for calculation, it may lead to an inaccurate calculation result. Therefore, C_{side} needs to prove that the data it uses for calculation are the latest on the blockchain. In this paper, we design a real-time proof generation method by leveraging a counting bloom filter to record the latest data stored on the chain.

Algorithm 1 Real-time verification

```
Input: B, B_s
Output: Real-time or Non-real-time
for i = 1; i ≤ m; i++ do
  if B_s[i] != 0 then
    if B[i] == 0 then
      return Non-real-time
    end if
  end if
end for
return Real-time
```

Bloom filter has good space efficiency and time efficiency, which is often used to verify whether an element exists in a set. However, a standard bloom filter cannot remove elements. This shortcoming can be compensated by applying a counting bloom filter (CBF) [29], in which each bit is replaced by a counter. It is incremented by one when the hash of the element hash is mapped to the counter. Conversely, when an element is removed, the counter is decremented by one.

We keep a m-bit set B in the block header of the latest generated block, which is composed of the latest data at that time by leveraging a counting bloom filter. When a cross-chain calculation task is performed, a new m-bit set B_s is generated synchronously by $user_s$, in which the data for calculation are mapped to B_s.

The set B stored in the latest block and the set B_s generated from this calculation are sent to SC for real-time verification. We design an audit algorithm on SC to perform bitwise operations on sets B and B_s, and the algorithm logic is shown in Algorithm 1. If there is a certain bit of 0, it indicates that the data for calculation are outdated; otherwise, we consider the data used for the calculation to be real-time.

5.5 Detailed Construction

Now, we give a detailed description of BeDCV. The main notations used in our scheme are displayed in Table 1 with brief explanations attached for ease of presentation.

TABLE 1
NOTATIONS AND DESCRIPTIONS

Notations	Descriptions
R	the calculation request from C_{main}
T	the timestamp of the latest block on C_{main}
q_1, q_2	large primes for paillier encryption
p	the large prime for bilinear pairing
N, g	the public keys of paillier encryption
λ, μ	the private keys of paillier encryption
u_i	the private keys for C_{side}
β_i, υ	the public parameters in this scheme
m_i	the data used for calculation
c_i	the ciphertext of m_i
M	the calculation result generated on C_{side}
C	the ciphtertext of the calculation result M on C_{side}
δ	the ciphertext result of the calculation on ciphertexts
B	the CBF in block header
B_s	the CBF of data involved in the calculation

Initialization: SC generates system parameters for consistency auditing and issues parameters to C_{side} and SC. The details are as follows:

- Select a large prime p and determine a bilinear pairing map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$, where $\mathbb{G}$, $\mathbb{G}_1$ are two multiplicative cyclic groups with the same order p.
- Publish the system parameters $\Omega = (e, \mathbb{G})$.

Task Requesting: When $user_m$ has a cross-chain calculation request, it gets the timestamp T from C_{main} and generates the private and public keys for paillier homomorphic encryption. Then, it sends the calculation request $R = (Task, T)$ and the public key N to SC and $user_m$ of C_{side}.

- Obtain the timestamp T from the latest generated block on C_{main}.
- Base on the security parameter κ , select three large primes q_1, q_2, and g. Then, compute the public key of paillier cryptosystem as $N = q_1 q_2$ and the private keys λ and μ.
- Send cross-chain calculation request $R = (Task, T)$ and the public key N to SC and $user_m$ of C_{side}, where $Task$ is the calculation that need to be done by $user_s$ of C_{side}.

Data calculation: $User_s$ completes the calculation task R by calling data $\{m_i\}_{i=1}^k$ from C_{side} to generate the result M. After calculation, it performs paillier homomorphic encrtption to process data $\{m_i\}_{i=1}^k$ into $\{c_i\}_{i=1}^k$ and generates authenticators of $\{c_i\}_{i=1}^k$ as $\{\sigma_i\}_{i=1}^k$. The data involved in the calculation are mapped into a new counting bloom filter B_s. Then, $user_s$ sends M, $\{c_i\}_{i=1}^k$, $\{\sigma_i\}_{i=1}^k$, $\{name_i\}_{i=1}^k$, $\{\beta_i\}_{i=1}^k$, N, h, υ, and B_s to C_{side}. The details of this procedure are as follows:

- Perform the calculation according to $Task$ to get the result M.
- Randomly choose $u_1, u_2, \ldots, u_k$ from $\mathbb{Z}_N^*$ and υ from $\mathbb{G}$, where k is the maximum number of data that can be calculated at the same time.
- Compute $\beta_1, \beta_2, \ldots, \beta_k$, where $\beta_i = T^{u_i} \in \mathbb{G}$, $i \in [1, k]$.
- Set two collision-resistant hash functions: $h : \{0,1\}^* \rightarrow \mathbb{Z}_{N^2}^*$ and $H : \{0,1\}^* \rightarrow \mathbb{G}$.
- Encrypt k data $m_1, m_2, \ldots, m_k$ with timestamp T as:

$$c_i = g^{m_i} \cdot h(T)^N \ (mod\ N^2). \tag{1}$$

- Build the counting bloom filter B_s with $\{m_i\}_{i=1}^k$.
- Compute an authenticator for c_i as:

$$\sigma_i = H(name_i || i)^{u_i} \cdot \upsilon^{c_i u_i}. \tag{2}$$

  Here $name_i \in Z_p^*$ is a identifier of the data m_i.
- Store M, $\{c_i\}_{i=1}^k$, $\{\sigma_i\}_{i=1}^k$, $\{name_i\}_{i=1}^k$, $\{\beta_i\}_{i=1}^k$, N, h, υ, and B_s into C_{side}.

Data Writing: C_{side} stores the information from $user_s$ into its block in the format of (key = T, value = $(M, \{c_i\}_{i=1}^k, \{\sigma_i\}_{i=1}^k, \{name_i\}_{i=1}^k, \{\beta_i\}_{i=1}^k, N, h, \upsilon, B_s)$). The smart contract S_s can call the value with the key T and calculate the corresponding audit proof with the determined audit proof generation algorithm. In addition, smart contract S_s encrypts M with paillier homomorphic encryption as C and transfers it to C_{main}. Then, C_{side} sends the audit proof π to SC. The details of this procedure are as follows:

- Encrypt result M with timestamp T as:

$$C = g^M \cdot h(T)^N \ (mod\ N^2). \tag{3}$$

- Compute the aggregated authenticator $\sigma = \prod_{i=1}^k \sigma_i$, as well as a random blind value $\varsigma = \prod_{i=1}^k \beta_i^{c_i}$.
- Send the encrypted result C to C_{main} and proof $\pi = (\{c_i\}_{i=1}^k, \sigma, \varsigma, \upsilon, \{\beta_i\}_{i=1}^k, \{name_i\}_{i=1}^k, B, B_s)$ to SC.

Results Receiving: Once C_{main} receives C, it stores C in its block and sends C to SC as the audit proof. $User_m$ can get C from C_{main} and decrypt it to get the plaintext M_r as follows:

$$M_r = L(C^{\lambda} (mod\ N^2)) \cdot \mu\ mod\ N. \tag{4}$$

Verification: the smart contract S on SC automatically performs the consistency verification according to the determined algorithm with the received audit proof from C_{side} and C_{main}. The details of this procedure are as follows:

- Perform the calculation on ciphertexts $\{c_i\}_{i=1}^k$ to get the calculation result δ.
- Compare C with δ. If it returns $true$, it means that the result of the cross-chain calculation request is calculated correctly by $user_s$.
- Perform the real-time verification algorithm with B and B_s. If it returns $true$, it means that $\{m_i\}_{i=1}^k$ involved in the calculation are up-to-date.
- Check the integrity verification equation:

$$e(\sigma, T) = \prod_{i=1}^{k} e(H(name_i||i), \beta_i) \cdot e(\upsilon, \varsigma) \quad (5)$$

  If the equation holds, it means that the audit proof provided by C_{side} is integral.

If all the verifications are passed, it means that the calculation result of the data provided by C_{side} is consistent.

6 Security Analysis

In this section, we briefly evaluate the security of BeDCV.

Theorem 1. *The privacy of the data involved in the calculation is preserved throughout the whole process if and only if the Decisional Composite Residuosity Assumption (DCRA) holds.*

Proof: Each data involved in the calculation are encrypted as $c_i = g^{m_i} \cdot h(T)^N (mod\ N^2)$. Even if the adversary $\mathcal{A}$ eavesdrops on data in cross-chain, it could only capture ciphertexts $\{c_i\}_{i=1}^k$. Since each c_i is actually a ciphertext of paillier cryptosystem, and $h(T)^N$ is used to play the role of random number in paillier cryptosystem, as analyzed in [16], [30], it achieves semantic security against the chosen plaintext attacks, and thus $\mathcal{A}$ could not recover any single type of data from c_i.

Theorem 2. *The integrity of the cross-chain interaction realizes if C_{side} can generate a correct proof of the data for calculation and the smart contract installed on SC can always pass (5).*

Proof: By simplifying (5), it can be seen that SC could always make a correct judgment about the integrity of data involved in the cross-chain calculation. The details are as follows:

$$\begin{aligned} LHS &= e(\prod_{i=1}^{k}\sigma_i, T) \\ &= e(\prod_{i=1}^{k} H(name_i||i)^{u_i} \cdot \upsilon^{c_i u_i}, T) \\ &= \prod_{i=1}^{k} e(H(name_i||i)^{u_i}, T) \cdot e(\upsilon^{\sum_{i=1}^{k} c_i u_i}, T) \\ &= \prod_{i=1}^{k} e(H(name_i||i), T^{u_i}) \cdot e(\upsilon, T^{\sum_{i=1}^{k} c_i u_i}) \\ &= \prod_{i=1}^{k} e(H(name_i||i), \beta_i) \cdot e(\upsilon, \prod_{i=1}^{k} \beta_i^{c_i}) \\ &= RHS \end{aligned} \quad (6)$$

Theorem 3. *Valid integrity proof (σ, ς) can only be generated when C_{side} faithfully transfers the audit data to SC.*

Proof: Suppose a probabilistic polynomial time adversary $\mathcal{A}$ can generate a forged proof (σ^*, ς^*), $(\sigma^*, \varsigma^*) \neq (\sigma, \varsigma)$ and try to pass the verification on SC.

We can get the following two equations:

$$e(\sigma, T) = \prod_{i=1}^{k} e(H(name_i||i), \beta_i) \cdot e(\upsilon, \varsigma) \quad (7)$$

$$e(\sigma^*, T) = \prod_{i=1}^{k} e(H(name_i||i), \beta_i) \cdot e(\upsilon, \varsigma^*) \quad (8)$$

Dividing (7) with (8), we can obtain:

$$\frac{e(\sigma, T)}{e(\sigma^*, T)} = \frac{e(\upsilon, \varsigma)}{e(\upsilon, \varsigma^*)} \quad (9)$$

Now we conduct a case analysis on (9).

Case 1: $\varsigma^* \neq \varsigma$

We assume the adversary $\mathcal{A}$ could successfully replace or tamper with a ciphertext c_i as c_i^*, thus $\triangle c_i = c_i - c_i^* \neq 0$, and $\varsigma^* = \beta_i^{c_i} \cdot \prod_{i' \neq i}^{k} \beta_{i'}^{c_{i'}}$, $\triangle \varsigma = \varsigma - \varsigma^* \neq 0$. Hence, the forged audit proof (σ, ς^*) could pass integrity verification equation. We can get the following equation from (9):

$$e(\upsilon, \varsigma) = e(\upsilon, \varsigma^*) \quad (10)$$

Since $\varsigma^* \neq \varsigma$, (10) will never hold. Therefore, it is computationally infeasible for $\mathcal{A}$ to pass the verification by tampering with the ciphertext of data with a non-negligible probability.

Case 2: $\sigma^* \neq \sigma, \varsigma^* \neq \varsigma$

We assume the adversary $\mathcal{A}$ could successfully replace or tamper with a ciphertext c_i as c_i^*, and even tamper with the corresponding authenticators σ_i as σ_i^*, thus $c_i^* \neq c_i$ and $\sigma_i^* \neq \sigma_i$.

If $\sigma^* = \sigma$, it follows from the verification equation that $\varsigma^* = \varsigma$, which contradicts to **case 1**. Therefore, no valid forged ς^* and σ^* where $\sigma^* \neq \sigma$ and $\varsigma^* \neq \varsigma$ that be found by the adversary $\mathcal{A}$ with non-negligible probability.

As a result, **theorem 3** is proved properly.

Theorem 4. *The real-time performance of the cross-chain interaction is guaranteed if C_{side} honestly performs the calculation with the latest data stored on C_{side}.*

Proof: If the adversary $\mathcal{A}$ passes the out-dated information m^* as real-time information m to generate the real-time audit proof, then $hash_i(m^*) \neq hash_i(m)$ $(i \in [1, n])$. The B_s generated according to $hash_i(m^*)$ cannot pass the real-time verification with B, which indicates that m^* is the non-real-time information.

Especially, since hash functions may have hash collisions, the bloom filter has a false positive rate, that is, the bloom filter reports that an element exists in a set, but in fact, the element is not in the set. We derive the false positive rate as follows:

Assuming that hash functions select and set a certain bit in the set with equal probability, m is the size of the set, and n is the number of hash functions, then the probability that a specific bit in the set is not set by hash operations while inserting an element is:

$$1 - \frac{1}{m}.$$

Then the probability that the bit is not set to "1" after all n hash operations is:

$$\left(1 - \frac{1}{m}\right)^n.$$

If we insert k elements, the probability that a bit is still "0" is:

$$\left(1 - \frac{1}{m}\right)^{kn}.$$

So the probability that this bit is "1" is:

$$1 - \left(1 - \frac{1}{m}\right)^{kn}.$$

If an element is not in the set, the probability that it is considered to be in the set is:

$$\left[1-\left(1-\frac{1}{m}\right)^{kn}\right]^{n} \approx \left(1-e^{-\frac{kn}{m}}\right)^{n}. \tag{11}$$

It is not difficult to see from the (11) in VI that as the size of the CBF array m increases, the false positives will decrease, and as the number of inserted data k increases, the false positives will also increase. For a given m and k, how to choose the number n of hash functions is determined as $n = ln2 \cdot mk^{-1}$. The false positives at this time is $P = 2^{-n}$. For a given false positive P, the optimal array size m of CBF can be derived as follows:

$$\begin{aligned} P &= 2^{-n} = 2^{-\frac{m}{k}ln2} \\ m &= -\frac{klnP}{(ln2)^2}. \end{aligned} \tag{12}$$

When we control the false positive at 0.05%, it can be calculated from (12) as $m \approx 11.03k$, which is the best ratio of the size of CBF and the amount of data while keeping the false positive low and efficiency high. Thus, we can get the most suitable number of hash functions as $n \approx 8$.

Theorem 5. *The correctness of the cross-chain interaction is guaranteed if the calculation result received by C_{main} is correctly calculated from the data on C_{side}.*

Proof: We set the calculation result of ciphertexts $\{c_i\}_{i=1}^{k}$ as $C = c_1 \cdot c_2 \cdot \cdots \cdot c_k$ and the plaintext of the calculation result as $M = m_1 + m_2 + \cdots + m_k$. Now, we deduce the decryption process of the ciphertext to verify the correctness of the calculation:

$$\begin{aligned} Dec(C) &= L(C^{\lambda}\ mod\ N^2) \cdot \mu \\ &= L((\textstyle\prod_{i=1}^{k} c_k)^{\lambda}\ mod\ N^2) \cdot \lambda^{-1} \\ &= L((g^{\lambda \sum_{i=1}^{k} m_i} \cdot h(T)^{kN})\ mod\ N^2) \cdot \lambda^{-1} \\ &= L((g^{M \cdot \lambda})\ mod\ N^2) \cdot \lambda^{-1} \\ &= \lambda \cdot M \cdot \lambda^{-1} \\ &= M \end{aligned} \tag{13}$$

(13) proves the correctness guarantee of paillier homomorphic encryption for data calculation. It achieves semantic security against the chosen plaintext attacks, thus $\mathcal{A}$ could not recover any information from ciphertexts c_i, which means that the proposed correctness verification method is effective.

Additionally, we assume that the ciphertext c_i is changed to c_i^* by adversary $\mathcal{A}$, where $c_i^* \neq c_i$, and try to pass the correctness verification. We can get $C^* = c_i \cdot \prod_{i' \neq i}^{k} c_{i'}$, where $C^* \neq C$.

From the simplification of (13), it can be seen that:

$$Decrypt(C^*) = \textstyle\sum_{i' \neq i}^{k} m_{i'} + Decrypt(c_i^*).$$

Since $c_i^* \neq c_i$, $Decrypt(c_i^*) \neq Decrypt(c_i)$, where $Decrypt(c_i) = m_i$. As a result, no valid forged c_i^*, where $c_i^* \neq c_i$, that be found by the adversary $\mathcal{A}$ to pass the correctness verification of the cross-chain calculation.

7 Performance Evaluation

In this section, we conduct a series of experiments to evaluate the performance of BeDCV.

7.1 Implementation and evaluation overview

We implement BeDCV based on the pairing-based library (PBC-0.5.14)[3] which performs the mathematical operations underlying pairing-based cryptosystems. Considering the interaction between multiple chains in BeDCV, we use the consortium blockchain to simulate the test environment. All the results of experiments are represented 100 trials on average.

To complete the experiments, we deploy three blockchains in Hyperledger Fabric 2.4[4]: a main chain, a side chain, and a supervision chain. All the on-chain procedures are leveraged by smart contracts installed on C_{side} and SC. The users off-chain use workstations running Ubuntu 16.04 with an Intel 2.5GHz CPU and 4GB memory.

Next, we will conduct experiments to evaluate the performance of BeDCV. The focus of cross-chain consistency verification is mainly on verifying the correctness of the calculation result, the integrity of audit proof, and the real-time performance of data.

7.2 Experiments and analysis

We conducted a series of experiments to evaluate the performance of BeDCV.

Communication cost in cross-chain auditing. SC receives audit proof from C_{side} and C_{main} for consistency verification. The size of the audit proof sent by C_{main} is a constant value, which is determined by the size of the ciphertext C. The audit proof sent by C_{side} is used to audit the correctness, integrity, and real-time performance of cross-chain interaction. The communication cost of cross-chain auditing is shown in Table 2. It can be seen that the communication cost is mainly concentrated in the correctness verification, while the cost of the real-time verification is a constant value. The overhead of correctness verification is mainly affected by the length of the ciphertext encrypted with paillier homomorphic encryption.

TABLE 2
Experimental Results of Communication Costs

k	Correctness(KB)	Integrity(KB)	Real-time(KB)
200	300.78	6.3	24.4
400	601.56	12.55	24.4
600	902.34	18.8	24.4
800	1203.13	25.05	24.4

The correctness of the calculation result. To audit the correctness of the calculation result of C_{side}, we need to perform paillier homomorphic encryption on the original data involved in the calculation to generate ciphertexts. During the audit, SC performs the homomorphic operation on ciphertexts sent by C_{side} to verify the correctness of the calculation result. We set the number of data involved in the calculation as k = 200, 400, 600, and 800 respectively. The detailed experimental results of the process of data for calculation are listed in Table 3, including the computational costs of paillier key generation (Com_{PKG}) and data encryption (Com_{DE}). For paillier homomorphic encryption,

3. https://crypto.stanford.edu/pbc/howto.html
4. https://hyperledger-fabric.readthedocs.io/en/release-2.4/

the time required for key generation is approximate and is not affected by the number of data involved in the calculation. The time for data encryption increases slightly as the number of data increases.

TABLE 3
Experimental Results of Data Processing

k	Com_{PKG}(ms)	Com_{DE}(s)
200	65.6	1.721
400	67.2	3.584
600	64.0	5.740
800	64.7	7.013

The integrity of audit proof. After encrypting the data involved in the calculation with paillier, the supervision node on C_{side} generates authenticators of ciphertexts and aggregates them into audit proof. The computational costs of key generation for integrity verification (Com_{KG}), authenticators generation (Com_{AG}), proof generation (Com_{PG}), and the bilinear pairing of integrity verification (Com_{KG}) are listed in Table x. We can conclude that the number of keys used for integrity verification is determined by the number of data involved in the operation. The time required for authenticator generation increases slightly with the number of ciphertexts and evidence generation is less affected by the number of data in milliseconds. In addition, the integrity verification takes a very short time and fluctuates less with the number of data involved in the calculation.

Once receiving proof from C_{side}, SC performs the integrity verification. We conducted experiments on SC about the computational costs of key generation for integrity verification (Com_{KG}) and the bilinear pairing of integrity verification (Com_V) and obtained experimental results in Table 4.

TABLE 4
Experimental Results of Integrity Verification

k	Com_{KG}(s)	Com_{AG}(s)	Com_{PG}(s)	Com_V(s)
200	0.247129	0.417376	0.013042	0.013042
400	0.458125	0.825037	0.013066	0.013065
600	0.688925	1.231268	0.014064	0.014064
800	0.910574	1.638940	0.014231	0.014226

The real-time of data. C_{side} needs to provide SC with real-time proof of data involved in the calculation to ensure that outdated data on C_{side} are not used for calculation. We utilize a counting bloom filter (CBF) to store the real-time information list in the latest block. SC audits the real-time performance of data according to the proof of CBF during cross-chain interaction.

The difference between BF and CBF is that every bit in BF is replaced by a counter, thus realizing the dynamic update of data. First, we conduct experiments on the computational costs of real-time proof generation using BF and CBF with k = 200, 400, 600, and 800 respectively when n=8. From Fig. 6, we can conclude that the generation time of the proof will increase slightly as the amount of inserted data increases. Besides, the calculation time of CBF will be longer than BF, since each insertion of CBD requires arithmetic calculation instead of setting the corresponding bit to $True$.

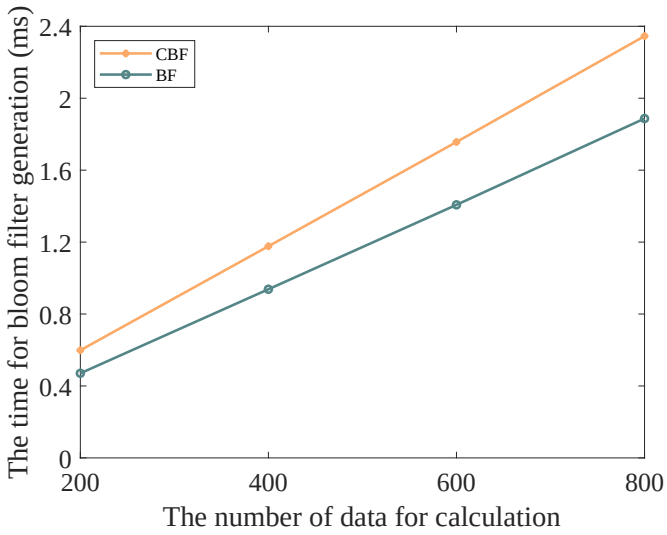

Fig. 6. The CBF Generation Time for CBF and BF

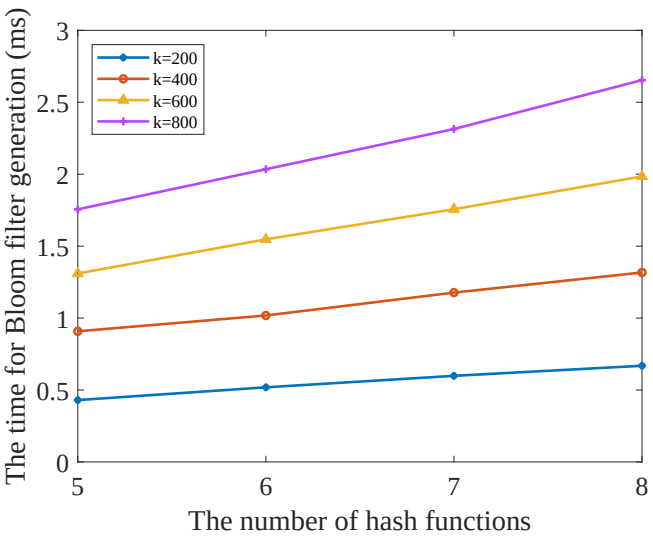

Fig. 7. The CBF Generation Time of Different Hash Times

Another factor that affects the generation of real-time proof is the number of hash functions in CBF. As can be seen from Fig. 7, the more hash functions used for data processing in CBF, the higher the total computational cost. Although the number of hash functions has an impact on the computational cost, the computational cost is slowly increasing in milliseconds, and overall it is still very time efficient.

Fig. 8 shows the computation costs of verifying the real-time proof on SC for different sizes of CBF when n=8. During verification, each bit of the CBF needs to be compared, so the larger the size of the CBF, the more time it takes to verify the proof.

8 Conclusion

In this paper, we propose a decentralized consistency verification scheme for cross-chain calculation (BeDCV) to relieve the storage and calculation pressure of single-chain, which introduces a supervision chain to conduct the consistency

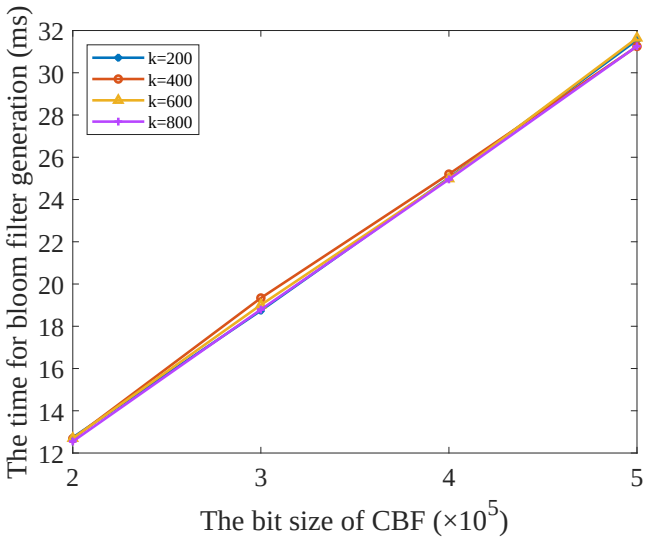

Fig. 8. The CBF Generation Time of Differen Size of CBF

audit on the result of cross-chain calculation. We utilize paillier homomorphic encryption, bilinear pairing, and counting bloom filter technologies to audit the correctness, integrity, and real-time performance of cross-chain calculation, which effectively ensures the consistency of the cross-chain calculation and realizes secure and reliable expansion of single-chain performance. Currently, the calculation auditing that we can achieve is limited to additional operations. We will continue to explore cross-chain auditing methods in future work, realizing consistency verification in various types of calculations.

REFERENCES

[1] A. Fox and E. Brewer, "Harvest, yield, and scalable tolerant systems," in *Proc. Workshop Hot Top. Oper. Syst. (HotOS)*, 1999, pp. 174–178.
[2] H. Bailie, Adrian, Thomas, and Stefan, "Interledger: Creating a standard for payments," in *Proc. Int. Conf. Companion World Wide Web (WWW)*, 2016, pp. 281–282.
[3] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. K. Miller, A. Poelstra, J. Timón, and P. Wuille, "Enabling blockchain innovations with pegged sidechains," *Sidechains protocol white paper*, 2014.
[4] L. Deng, H. Chen, J. Zeng, and L. Zhang, "Research on cross-chain technology based on sidechain and hash-locking," in *Proc. Int. Conf. Edge Comput. (EDGE)*, 2018, pp. 144–151.
[5] Y. He, C. Zhang, B. Wu, Y. Yang, K. Xiao, and H. Li, "A cross-chain trusted reputation scheme for a shared charging platform based on blockchain," *IEEE Internet Things J.*, pp. 1–1, 2021.
[6] ——, "Cross-chain trusted service quality computing scheme for multi-chain model-based 5g network slicing sla," *IEEE Internet Things J.*, pp. 1–1, 2021.
[7] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in *Proc. Eur. Symp. Res. Comput. Secur. (ESORICS)*, 2009, pp. 355–370.
[8] H. Wang, Q. Wang, and D. He, "Blockchain-based private provable data possession," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 5, pp. 2379–2389, 2021.
[9] H. Jin, H. Jiang, and K. Zhou, "Dynamic and public auditing with fair arbitration for cloud data," *IEEE Trans. Cloud Comput.*, vol. 6, no. 3, pp. 680–693, 2018.
[10] Y. Zhang, C. Xu, X. Lin, and X. Shen, "Blockchain-based public integrity verification for cloud storage against procrastinating auditors," *IEEE Trans. Cloud Comput.*, vol. 9, no. 3, pp. 923–937, 2021.
[11] Y. Du, H. Duan, A. Zhou, C. Wang, M. H. Au, and Q. Wang, "Enabling secure and efficient decentralized storage auditing with blockchain," *IEEE Trans. Dependable Secure Comput.*, pp. 1–12, 2021.
[12] D. Li, J. Liu, Z. Tang, Q. Wu, and Z. Guan, "Agentchain: A decentralized cross-chain exchange system," in *Proc. IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun./IEEE Int. Conf. Big Data Sci. Eng. (TrustCom/BigDataSE)*, 2019, pp. 491–498.
[13] W.-T. Tsai, R. Blower, Y. Zhu, and L. Yu, "A system view of financial blockchains," in *Proc. IEEE Symp. on Service-Oriented Syst. Eng. (SOSE)*, 2016, pp. 450–457.
[14] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, pp. 120–126, 1978.
[15] L. Chen, Y. Xu, W. Fang, and C. Gao, "A new elgamal-based algebraic homomorphism and its application," in *Proc. ISECS Int. Colloquium Comput., Commun., Control, Management (CCCM)*, vol. 1, 2008, pp. 643–648.
[16] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptographic Tech. (EUROCRYPT)*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 223–238.
[17] X. Zhang, C. Huang, Y. Zhang, and S. Cao, "Enabling verifiable privacy-preserving multi-type data aggregation in smart grids," *IEEE Trans. Dependable Secure Comput.*, pp. 1–1, 2021.
[18] H. Bao and R. Lu, "A new differentially private data aggregation with fault tolerance for smart grid communications," *IEEE Internet Things J.*, vol. 2, no. 3, pp. 248–258, 2015.
[19] X. Zhang, C. Xu, H. Wang, Y. Zhang, and S. Wang, "Fs-peks: Lattice-based forward secure public-key encryption with keyword search for cloud-assisted industrial internet of things," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 3, pp. 1019–1032, 2021.
[20] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," *Electron. Colloq.on Comput. Complexity (ECCC)*, vol. 18, p. 111, 01 2011.
[21] L. Ducas and D. Micciancio, "Fhew: Bootstrapping homomorphic encryption in less than a second," in *Annu. Int. Conf. Theory Appl. Crypt. Tech. (EUROCRYPT)*, 2015, pp. 617–640.
[22] N. P. Smart and F. Vercauteren, "Fully homomorphic simd operations," *Des., Codes and Crypt.*, vol. 71, no. 1, p. 57–81, Apr. 2014.
[23] W. Licheng and L. Jing, "Simple analysis on noiseless fully homomorphic encryptions," *Journal of Crypt. Res.*, vol. 4, no. 6, p. 579, 2017. [Online]. Available: http://www.jcr.cacrnet.org.cn/CN/abstract/abstract228.shtml
[24] A. Sutton and R. Samavi, "Blockchain enabled privacy audit logs," in *Proc. 16th Int. Semantic Web Conf. (ISWC)*, 2017, pp. 645–660.
[25] X. Liang, S. Shetty, D. Tosh, C. Kamhoua, K. Kwiat, and L. Njilla, "Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability," in *Proc. 17th IEEE/ACM Int. Symp. Cluster Cloud Grid Comput. (CCGRID)*, 2017, pp. 468–477.
[26] A. Menezes, "An introduction to pairing-based cryptography," 2005. [Online]. Available: http://www.math.uwaterloo.ca/∼ajmeneze/publications/pairings.pdf
[27] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *Annu. Int. Conf. Theory Appl. Crypt. Tech. (EUROCRYPT)*, 2010, pp. 24–43.
[28] S. Tarkoma, C. E. Rothenberg, and E. Lagerspetz, "Theory and practice of bloom filters for distributed systems," *IEEE Commun. Surveys Tutorials*, vol. 14, no. 1, pp. 131–155, 2012.
[29] L. Fan, P. Cao, J. Almeida, and A. Broder, "Summary cache: a scalable wide-area web cache sharing protocol," *IEEE/ACM Trans. Netw.*, vol. 8, no. 3, pp. 281–293, 2000.
[30] S. Li, K. Xue, Q. Yang, and P. Hong, "Ppma: Privacy-preserving multisubset data aggregation in smart grid," *IEEE Trans. Ind. Informatics*, vol. 14, no. 2, pp. 462–471, 2018.

Yushu Zhang (Member, IEEE) received the B.S. degree from the School of Science, North University of China, Taiyuan, China, in 2010, and the Ph.D. degree from the College of Computer Science, Chongqing University, Chongqing, China, 2014. He held various research positions with City University of Hong Kong, Southwest University, University of Macau, and Deakin University. He is currently a Professor with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China. He is the Associate Editor of Information Sciences and Signal Processing. His research interests include multimedia security, blockchain, and artificial intelligence. He has authored or coauthored more than 100 refereed journal articles and conference papers in these areas.

Jiajia Jiang received the B.S. degree in network engineering from the School of Information Science and Technology, Nanjing Agricultural University, Nanjing, China, in June 2020. She is currently pursuing the M.S. degree with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China. Her research interests include blockchain and cryptography.

Xuewen Dong (Member, IEEE) received the B.E., M.S., and Ph.D. degrees in computer science and technology from the Xidian University of China, Xi'an, China, in 2003, 2006, and 2011, respectively. From 2016 to 2017, he was a Visiting Scholar with the Oklahoma State University, Stillwater, OK, USA. He is currently a Professor with the School of Computer Science and Technology, Xidian University, Xi'an, China. His research interests include cognitive radio network, wireless network security, and blockchain.

Liangmin Wang (Member, IEEE) received the B.S. degree in computational mathematics from Jilin University, Changchun, China, in 1999, and the Ph.D. degree in cryptology from Xidian University, Xi' an, China, in 2007. He is currently a Full Professor with the School of Cyber Science and Engineering, Southeast University, Nanjing, China. He has authored or coauthored more than 60 technical papers at premium international journals and conferences, such as the IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE Global Communications Conference, and IEEE Wireless Communications and Networking Conference. His current research interests include data security and privacy. He has served as a TPC member of many IEEE conferences, such as IEEE ICC, IEEE HPCC, and IEEE TrustCOM. He is currently an Associate Editor of Security and Communication Networks, a member of ACM, and a Senior Member of Chinese Computer Federation. He has been honored as a "Wan-Jiang Scholar" of Anhui Province since November 2013.

Yong Xiang (Senior Member, IEEE) received the Ph.D. degree in electrical and electronic engineering from the University of Melbourne, Australia. He is a Professor with the School of Information Technology, Deakin University, Australia. He has published six monographs, over 190 refereed journal articles, and numerous conference papers in his research areas. His research interests include information security and privacy, signal and image processing, data analytics and machine intelligence, the Internet of Things, and blockchain. He has served as the honorary chair, the general chair, the program chair, the TPC chair, the symposium chair, and the track chair for many conferences, and was invited to give keynotes at a number of international conferences. He was the Associate Editor of IEEE Signal Processing Letters and IEEE Access, and the Guest Editor of IEEE Transactions on Industrial Informatics and IEEE Multimedia. He is the Senior Area Editor of IEEE Signal Processing Letters and the Associate Editor of IEEE Communications Surveys and Tutorials.