IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 6 September 2022, accepted 27 September 2022, date of publication 3 October 2022, date of current version 11 October 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3211391

RESEARCH ARTICLE

A Blockchain-Based Efficient Data Integrity Verification Scheme in Multi-Cloud Storage

YIRAN ZHANG, HUIZHENG GENG, LI SU, AND LI LU
China Mobile Research Institute, Beijing 100053, China

Corresponding author: Huizheng Geng (genghuizheng@chinamobile.com)

This work was supported by the Beijing University of Posts and Telecommunications-China Mobile Research Institute Joint Innovation Center.

ABSTRACT The cloud storage service provides the storage and access function for massive data, reducing the management cost for large amounts of data. The data integrity verification scheme in cloud storage can be employed to help users confirm the integrity of outsourced data. Although public data integrity verification schemes allow users to outsource data integrity verification to third-party auditor (TPA), there are still many problems with centralized TPA in terms of security and efficiency. In recent years, researchers have tried to apply blockchain technology to solve the centralization problem of traditional methods, but these schemes do not pay attention to the problem of efficiency degradation caused by the use of blockchain technology. This paper proposes an efficient data integrity verification scheme for multi-cloud storage services by using blockchain technology. The overall verification can verify the integrity of multiple CSPs, which solves the problems of low computational efficiency. Local verification can trace the source to the specific damaged CSP, which is more secure and reliable. In addition, this paper puts the data verification process directly in the blockchain for public execution and provides data integrity verification services without the assistance of any third-party audit platform, avoiding the security problems caused by untrusted TPA. Theoretical analysis and experiments verify the safety and effectiveness of the scheme.

INDEX TERMS Cloud computing security, data integrity, blockchain, data security.

I. INTRODUCTION

With the rapid development of network and communication technology, cloud computing has been widely applied in recent years. Cloud storage is a service provided by cloud computing [1], which allows users to access the network and use storage resources. The cloud storage service provides the storage and access function for massive data, reducing the management cost of users for large amounts of data. Whether it is a single file or a large number of files, cloud storage services can make it easy for users to share files. Cloud storage services greatly reduce the burden of cloud users, but also bring some security risks to cloud data. On the one hand, compared with traditional storage methods, data stored in the cloud may be lost or damaged due to the damage of attackers or hardware and software failures, such as the outsourced data integrity damage event of Tencent Cloud in 2018. On the other hand, cloud service provider (CSP) may be untrustworthy. It may delete some data that users rarely or almost never access in order to obtain greater benefits, while it may hide the data loss accident from users to preserve its reputation. Since the user loses physical control over the data on the cloud, it is impossible to determine whether the data on the cloud is corrupted. Therefore, users need a cloud data integrity detection mechanism to verify whether the outsourced data is damaged.

The traditional method of data integrity verification is to download all the data of the data owner (DO) directly from the CSP and check the integrity of the data locally. However, it is impractical for data owners to store data in a large number of CSP clusters and download all data to the local machine during each integrity check, because this method wastes a large amount of network transmission resources and local storage resources, which seriously weakens the advantages

The associate editor coordinating the review of this manuscript and approving it for publication was Amjad Mehmood.

of cloud services. Due to the high cost and insufficient resources of local verification, a trusted third-party institution or mechanism is required to perform the verification task, so local verification is gradually replaced by public verification.

In the public verification schemes, the user entrusts a third-party auditor (TPA) to carry out the data integrity verification. In particular, TPA challenges the CSP and requires it to prove the integrity of the outsourced data, and then TPA sends the integrity verification report to DO. Although the participation of the centralized TPA seems to make the audit process more convenient, there are still many problems in terms of security and efficiency. For example, if TPA knows that the public audit process is conducted periodically, TPA may avoid costs by not performing several verification processes after one verification without problems, but will produce a report without problems; TPA may collude with the CSP, such as jointly concealing data corruption from users, or only verifying data blocks with good integrity; TPA is not controlled by the user and may leak data, etc.

At present, blockchain technology can provide a distributed database system, which has the potential to build a trusted, fair and decentralized environment for cloud storage [2], [3]. Since the blockchain records all transactions, few people can make any changes to the data once it is in the blockchain, making it difficult to forge, trace and tamper with. In recent years, researchers have designed some data audit schemes for data integrity verification based on blockchain [4], [5]. Some studies use blockchain to implement trusted storage of TPA audit logs which help users monitor untrusted TPAs [6], [7]. On this basis, researchers further use blockchain to replace TPA for trusted audit [8]. Xue et al. [9] proposed an identity-based public auditing (IBPA) scheme for cloud storage systems using the public blockchain mechanism of the Bitcoin system, but the resistance to malicious auditors increases the computational overhead on the user side. Zhang et al. [10] proposed a data audit scheme for multi-cloud storage based on blockchain, but the verification efficiency of this scheme was low. Hence, the use of blockchain technology will affect the efficiency of integrity verification with varying degrees, and even lead to the problem of low efficiency.

In order to solve the above problems, this paper proposes a blockchain-based data integrity verification scheme in multi-cloud storage. Since the information on the blockchain can be regarded as immutable and traceable, this paper puts the data verification process directly in the blockchain for public execution and provides data integrity verification services without the assistance of any third-party audit platform, avoiding the security problems caused by untrusted TPA. In addition, the integrity verification of multiple CSPs for multiple DOs is realized by the overall verification, which solves the problems of low computational efficiency. The specific CSP with integrity damage is traced by local verification, which solves the problem of tracing malicious CSPs in distributed cloud storage.

The main research work of this paper is as follows.

1) This paper proposes an efficient data integrity verification scheme for multi-cloud storage services using blockchain technology. Compared with centralized methods based on TPA, this scheme uses blockchain technology to achieve decentralized integrity verification services, which are more difficult to tamper with.

2) This scheme stores the aggregate signatures and local signatures of encrypted data in the blockchain, and the verification process is carried out publicly in the blockchain. Therefore, anyone in the blockchain network can serve as a public auditor, and the verification results can be repeatedly authenticated by them, resulting in the extensive recognized of verification results. In addition, the blockchain will publish the CSP with missing data integrity to supervise the CSP to guarantee the data integrity.

3) We designed two data integrity verification modes: overall verification and local verification. The overall verification can verify the integrity of multiple CSPs, which can improve the efficiency of integrity verification and resist rough key attack. Local verification can trace the source to the specific damaged CSP, which is more secure and reliable.

II. RELATED WORKS

In order to protect the integrity of outsourced data, the previous researches consider checking the MAC value of data in data audit, but these schemes have limited verification time and high overhead [11]. Ateniese et al. [12] first proposed the concept of PDP. In their RSA-based PDP model, by randomly checking part of the data stored in CSP, users can obtain probabilistic proof of data integrity through challenge response without downloading the complete file. After that, more PDP schemes were proposed [13]. Wang et al. [14] proposed a PDP protocol, which supports public data audit based on the trusted TPA to reduce the overhead in the process of data audit. Wang et al. [15] introduced ring signature technology to protect the privacy of user in the process of public audit. Yu et al. [16] proposed an identity-based data auditing mechanism to protect the privacy of user and reduce computational costs. However, most schemes usually rely on trusted TPA, which may not exist in distributed cloud storage environments and faces problems such as performance bottlenecks and single points of failure. The few methods that do not require TPA can only help the user discover that the outsourced data has been corrupted, but cannot help the user provide an accepted proof of integrity breach, which gives the CSP an opportunity to refuse to validate the results.

The rise of blockchain technology makes it possible to solve the problem of centralization of data audit schemes. A blockchain is an immutable distributed ledger that records the status of all entities in the blockchain network [2], [3]. With the characteristics of decentralization, non-repudiation and traceability of blockchain, many data auditing schemes based on blockchain have been proposed. Xue et al. [9] constructed random challenge messages through blockchain and implemented an identity-based public audit scheme to

prevent untrusted TPA and storage service providers from colluding and forging data integrity certificates. To supervise the semi-trusted TPA, Zhang et al. [17] required them to publish audit logs on the blockchain. Zhang et al. [18] also designed a similar data audit scheme based on blockchain. Zhang et al. [10] proposed a data audit scheme for multi-cloud storage based on blockchain, which introduced blockchain technology and realized the integrity audit of public batch data in multi-cloud scenarios, but the verification efficiency of this scheme was low. Although these solutions solve the centralization problem of traditional methods, they do not pay attention to the problem of efficiency degradation caused by the use of blockchain technology, especially in the CSP system containing multiple servers, the data integrity verification process will be extremely slow and even affecting the use of the data.

III. PRELIMINARY

A. PEDERSEN COMMITMENT

Cryptographic commitment scheme is a two-phase interaction protocol involving two parties. One of the most widely used cryptography commitment in privacy protection schemes is Pedersen commitment [19], which plays an indispensable role in numerical hiding, identity verification and audit verification [20]. Pedersen commitment based on elliptic curve consists of Committer and Verifier, including commitment generation phase and commitment disclosure phase:

Commitment generation: the committer chooses a random number r and computes the commitment $com\,(m) = m * F + v * H$ on the secret data m, where F and H are two fixed points with different positions on the elliptic curve, sends the commitment $com\,(m)$ to the verifier, and discloses F and H.

Commitment disclosure: the committer sends (m, v) to the verifier, and the verifier recalculates $com'\,(m)$ to verify whether $com\,(m)$ is equal to $com'\,(m)$, thus determine whether the data m has been tampered.

The Pedersen commitment has homomorphism. For $m_0, m_1 \epsilon G$ and $v_0, v_1 \epsilon G$ it has $com\,(m_0; v_0) + com\,(m_1; v_1) = (m_0 + m_1) * F + (v_0 + v_1) * H = com\,(m_0 + m_1; v_0 + v_1)$, where $com\,(m_0 + m_1; v_0 + v_1)$ indicates aggregate commitment, $com\,(m_0; v_0)$ and $com\,(m_1; v_1)$ indicates commitment.

B. MUSIG

Musig is a multi-signature scheme proposed by the Blockstream team in 2018 [21]. Multi-signature indicates that multiple signers jointly sign the same message. Musig aggregates the public keys of multiple signers into one public key (aggregated public key), sends the data to each signer for simultaneous signature, and aggregates all signatures into a total signature (aggregated signature), so the blockchain can directly use the aggregated public key to check the validity of the aggregated signature. Compared with the other multi-signature methods, Musig has two obvious advantages: ① it has the same key length and signature length as the standard Schnorr signature, which is very efficient; ② it ensures the security of Rogue Key attacks.

C. BLOCKCHAIN

A blockchain can be regarded as a distributed shared ledger built on a network, where the content in the blockchain is agreed upon by all nodes through a consensus mechanism. Therefore, the blockchain is very robust, and the error of a few entities will not affect the correct operation of the whole system. In addition, the distributed node authentication and negotiation mechanism is used to solve the trust problem between untrusted nodes in distributed information systems, so as to ensure the trust between untrusted distributed entities without the need for traditional trusted third party. Therefore, blockchain is very suitable to solve the centralized problem of trusted TPA, achieve decentralized data integrity verification, and make verification results more reliable.

IV. PROBLEM FORMULATION

A. SYSTEM MODEL

The system model in this paper includes three entities: data owner (DO), cloud server provider (CSP), and blockchain.

Data Owner owns a large amount of data and outsources them to CSP. The number of DOs is $n(n \geq 2)$. In the signature generation phase, data owners encrypt the original data with the homomorphic encryption technology, generate commitments, and upload them to the corresponding CSP respectively. In addition, the random numbers used to generate the commitment will be uploaded to the blockchain.

Cloud server provider possesses massive storage resources and computing resources. It is mainly responsible for storing encrypted data uploaded by the data owner and assisting with integrity verification. The number of CSPs is $n(n \geq 2)$. In the signature generation phase, CSPs independently generate local signatures, collaboratively generate aggregated signatures, and upload them to the blockchain. In the signature verification phase, CSPs collaboratively generate aggregate commitment and send it to the blockchain for overall verification, while CSPs directly generate commitments and send them to the blockchain for local verification.

Blockchain is mainly responsible for storing the aggregated signatures and local signatures uploaded by CSPs and the random number uploaded by DOs, and publicly verifying the integrity of the data. Blockchain is allowed to be accessed by data owners and authorized users, and the process and results of integrity verification will be checked repeatedly, so blockchain can guarantee the credibility of verification results.

Figure 1 represents the system model of ''A Blockchain-based data integrity verification scheme in cloud storage''. In the signature generation stage, ① DO1∼DOn encrypt data, generate commitments, and upload them to the corresponding CSP1∼CSPn respectively. In addition, the random number used to generate commitments can be uploaded to block1∼blockn of the blockchain. ② For overall verification,

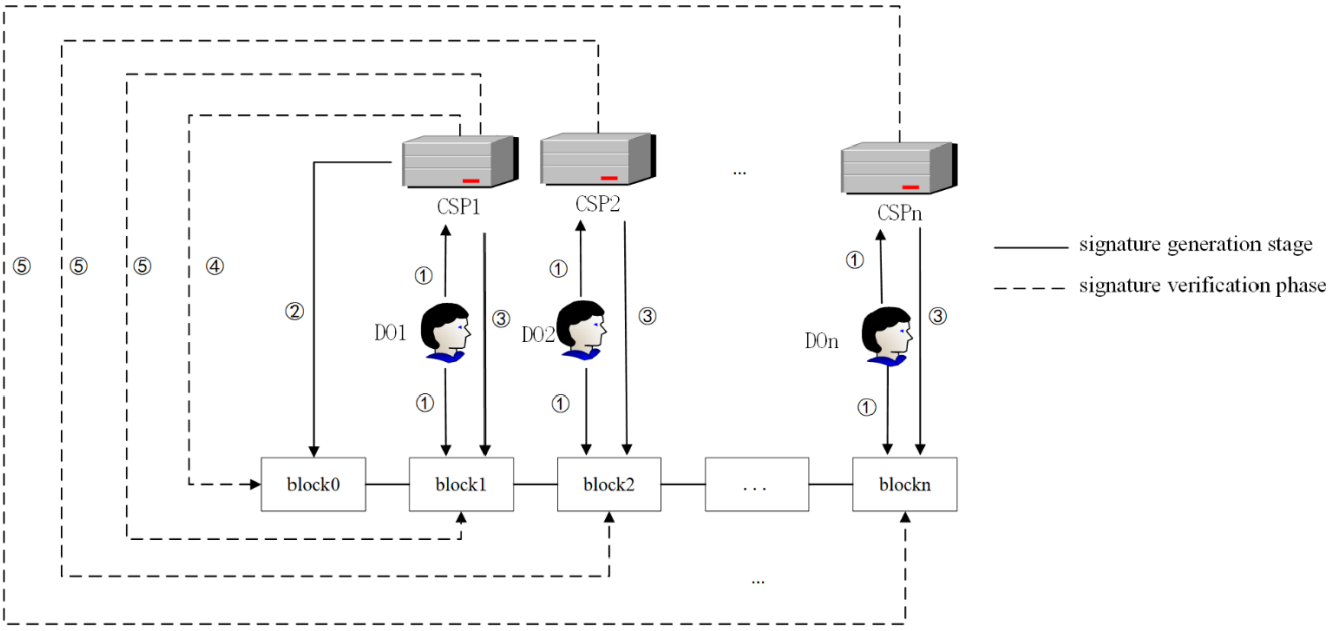

FIGURE 1. The system model of the scheme.

CSPs generate the aggregate signature collaboratively and uploads it to block0 of the blockchain. ③ For local verification, CSP1∼CSPn generate local signatures independently and upload them to block1∼blockn of the blockchain respectively. In the signature verification phase, ④ For overall verification, CSPs generate the aggregate commitment collaboratively and sends it to block 0 which verifies the integrity of the data on all servers. If the output is 1, it indicates that data on CSPs are not damaged; if the output is 0, it indicates that the data on one or more servers are damaged. ⑤ For local verification, CSP1∼CSPn generate local commitments of their stored data respectively and send them to corresponding block1∼blockn of the blockchain. If the output is 1, it indicates that the server is not damaged; if the output is 0, it indicates that the server is damaged. The damaged CSP will be disclosed on the blockchain.

B. THREAT MODEL

Assuming that all CSPs are not fully trusted that the malicious or careless CSP may make hardware/software errors, remove data rarely used to reduce storage burden, and try to hide the fact to maintain their reputation. The DOs can be trusted which strictly enforce the protocol and ensure the security of data. The blockchain can be trusted that most entities in the blockchain network are reliable and the information recorded in the blockchain is correct and publicly available.

C. SECURITY GOAL

1) EFFICIENCY

The low communication and computational overhead of the system should be ensured that the results can be get immediately after sending the verification request. This paper designs the overall verification that can verify the integrity of multiple CSPs to improve the efficiency of integrity verification.

2) PUBLIC

The integrity verification process should be fully public. The verification phase is carried out publicly on the blockchain that all authorized users and DOs could access to the blockchain and recalculate the verification process, resulting in the verification results can be widely recognized. In addition, the blockchain will publish CSPs with missing data in order to supervise CSPs to guarantee data integrity. The function can effectively avoid disputes over data integrity results by CSPs.

3) DATA PRIVACY

CSPs and blockchain cannot obtain plaintext information of users. In order to protect the privacy and security of sensitive information, data should be encrypted before outsourcing to CSPs. In addition, data uploaded to the blockchain cannot reveal any plaintext data.

V. THE PROPOSED SCHEME

This section describes the algorithm of a blockchain-based data integrity verification scheme in cloud storage, including overall verification and local verification. TABLE 1 shows some important notations.

A. OVERALL VERIFICATION

The overall verification includes the signature generation phase and the signature verification phase. In the signature generation phase, ① DOs encrypt data, generate the commitment, and upload them to the corresponding CSP

TABLE 1. Notations.

Notation	Description
n	the number of DOs/CSPs
m_i	the data of DOi($i\epsilon[1,n]$)
$commit_i$	In the signature generation phase of overall verification, the commitments generated by CSPi
$Sumcommit$	In the signature generation phase of overall verification, the aggregate commitments generated by CSPs
s_i	In the signature generation phase of overall verification, the signature generated by CSPi
s	In the signature generation phase of overall verification, the aggregate signature generated by CSPs
$Sumcommit'$	In the signature verification phase of overall verification, the aggregate commitments generated by CSPs
ls_i	In the signature generation phase of local verification, the local signature generated by CSPi
$lcommit$	In the signature verification phase of local verification, the commitments generated by CSPi

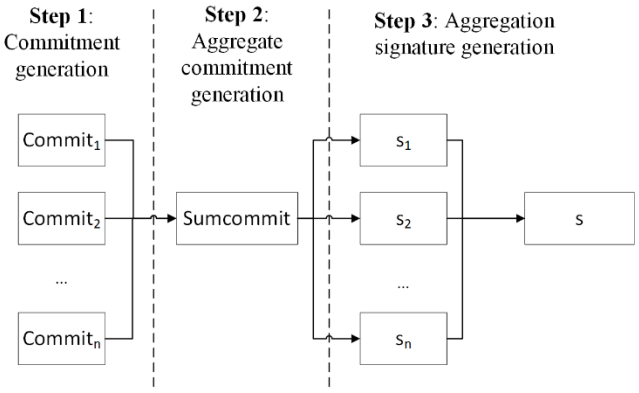

FIGURE 2. The flowchart of the signature generation phase for overall verification.

respectively. ② CSPs collaborate to generate aggregate commitment. ③ CSPs collaborate to generate aggregate signature with the Musig technology. In the signature verification phase, ④ CSPs generates the aggregate commitment and uploads it to block0 of blockchain. ⑤ Block0 verifies the signature, if the output is 1, it indicates that all data are not damaged; if the output is 0, it indicates that the data on one or more servers are damaged.

1) SIGNATURE GENERATION PHASE

Figure 2 shows the flowchart of the signature generation phase for overall verification. The detailed steps are as follows:

Step 1: Commitment generation $commit_i \leftarrow CommitGen(m_i)$.

The DOi encrypts the data m_i with the data encryption algorithm (such as the AES algorithm), gets the encrypted data $E(m_i)$, generates the commitment $commit_i$, and uploads it to the corresponding CSP i respectively.

$$commit_i = E(m_i) * F + v_i * H \tag{1}$$

where public information F and H are two fixed points with different positions on the elliptic curve in a finite field, $E(m_i)$ is the encrypted data of DOi, v_i is the random number generated by DOi which is uploaded to blocki.

Step 2: Aggregate commitment generation $Sumcommit \leftarrow SumcommitGen(commit_i)$.

CSPs exchange commitment to generate aggregate commitment.

$$Sumcommit = \sum_{i=1}^{n} commit_i \tag{2}$$

Step 3: Aggregation signature generation $\sigma \leftarrow Sig(Sumcommit)$.

Algorithm 1 shows the process of the Aggregation signature generation, and the detailed steps are as follows:

1. Parameter generation. Define the group elements (G, p, g), p is an integer of k bits, g is a generator of the group G, three hash functions $H_{com}, H_{agg}, H_{sig}$, where H_{com} is used to calculate t, H_{agg} is used to calculate the aggregation key, and H_{sig} is used to generate the signature.
2. Key generation. CSPi ($i \in [1, \ldots, n]$) randomly selects a random number as the private key $sk_i = x_i \leftarrow Z_p$, generates the corresponding public key $pk_i = X_i = x_i * g$.
3. Aggregate public key generation. CSPi calculates $a_i = H_{agg}(L, X_i)$, in which $L = \{PK_1 = X_1, \ldots, PK_n = X_n$ denotes the set of public keys; CSPi calculates the aggregate public key $\tilde{X} = \sum_{i=1}^{n} (a_i * X_i)$.
4. Aggregate signature generation.
   a) CSPi ($i \in [1, \ldots, n]$) select a random number $r_i \leftarrow Z_p$, calculate

$$R_i = r_i * g \tag{3}$$
$$t_i = H_{com}(R_i) \tag{4}$$

   and send t_i to the other CSPs.
   b) After receiving $t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_n$ from the other CSPs, CSPi sends R_i to the other CSPs.
   c) After receiving $R_1, \ldots, R_{i-1}, R_{i+1}, \ldots, R_n$ from the other CSPs, CSPi ($i \in [1, \ldots, n]$) checks $t_{i'} = H_{com}(R_{i'})$ for $i' \in [1, \ldots, i-1, i+1, \ldots, n]$, and aborts the protocol if this is not the case; otherwise, it computes $R = \sum_{i=1}^{n} R_i$ and send R_i to block i.
   d) CSPi computes the signature

$$s_i = r_i + ca_i x_i \ mod \ p \tag{5}$$

where $c = H_{sig}(\tilde{X}, R, Sumcommit)$ and sends s_i to CSP1.

e) After receiving $s_2, \ldots, s_n$ from the other CSPs, CSP1 calculates $s = \sum_{i=1}^{n} s_i$. CSP1 generates the aggregate signature $\sigma = (s, R)$ and sends it to block 0.

Algorithm 1 $(\sigma) \leftarrow Sign\left((sk, pk), L, Sumcommit\right)$

Input: (sk, pk), L, $Sumcommit$
Output: σ
1: **For** $i = 1 : n$
2: CSPi selects $r_i \leftarrow Z_p$, calculates $R_i = r_i * g$, $t_i = H_{com}(R_i)$, sends t_i to the other CSPs;
3: **End**
4: **For** $i = 1 : n$
5: CSPi sends R_i to the other CSPs;
6: **End**
7: **For** $i = 1 : n$
8: $R = 0$;
9: **For** $i' = 1 : n$ && $i' \neq i$
10: **If** $t_{i'} == H_{com}(R_{i'})$ **then**
11: $R = R + R_{i'}$;
12: **Else**
13: Break;
14: **End**
15: **End**
16: **For** $i = 1 : n$
17: CSPi computes the signature $s_i = r_i + ca_i x_i \ mod\ p$, and sends s_i to CSP1;
18: **End**
19: CSP1 calculates $s = \sum_{i=1}^{n} s_i$;
20: **Return** $\sigma = (s, R)$

2) SIGNATURE VERIFICATION PHASE

Algorithm 2 shows the process of the signature verification phase of overall verification, and the detailed steps are as follows:

Step 1: Aggregate commitment creation $Sumcommit' \leftarrow SumcommitCreate(Enc\left(\hat{m}_i\right))$.

CSPs compute $\sum_{i=1}^{n} Enc(\hat{m}_i)$, where $Enc(\hat{m}_i)$ indicates the encrypted data stored in the CSP i.

CSP1 takes $v = \sum_{i=1}^{n} v_i$ from the blockchain, generates the aggregate commitment $Sumcommit'$ and sends it to block0.

$$Sumcommit' = \sum_{i=1}^{n} Enc(\hat{m}_i) * F + v * H \tag{6}$$

Step 2: Aggregate signature verification $0/1 \leftarrow verification(Sumcommit', L, \sigma)$.

Blockchain stores a set of public keys $L = \{X_1, \ldots, X_n\}$, aggregate signature $\sigma = (R, s)$, aggregate commitment $Sumcommit'$, block0 calculates $a_i = H_{agg}(L, X_i)$, $\tilde{X} = \sum_{i=1}^{n} (a_i * X_i)$, $c' = H_{sig}(\tilde{X}, R, Sumcommit')$, verifies that $s * G = R + c'\tilde{X}$. if the output is 1, it indicates that all data are not damaged; if the output is 0, it indicates that the data on one or more servers are damaged and local verification can be performed to find the compromised server.

Algorithm 2 Signature Verification Phase of Overall Verification

Input: $Enc\left(\hat{m}_i\right)$, L, $\sigma = (s, R)$
Output: 0/1
1: $SumEnc = 0$
2: **For** $i = 1 : n$
3: CSPi calculates $SumEnc = SumEnc + Enc\left(\hat{m}_i\right)$;
4: **End**
5: CSP1 calculates $Sumcommit' = SumEnc * F + v * H$;
6: **For** $i = 1 : n$
7: Block0 calculates $a_i = H_{agg}(L, X_i)$;
8: **End**
9: Block0 calculates $\tilde{X} = \sum_{i=1}^{n} (a_i * X_i)$, $c' = H_{sig}\left(\tilde{X}, R, Sumcommit'\right)$;
10: **If** $s * G == R + c'\tilde{X}$ **then**
11: $flag = 1$;
12: **Else**
13: $flag = 0$;
14: **Return** $flag$

B. LOCAL VERIFICATION

Local verification includes signature generation phase and signature verification phase. In the signature generation phase, CSPi generates the local signature and sends it to blocki of the blockchain. In the signature verification phase, ① CSPi generates a local commitment for the stored data and sends it to blocki of the blockchain; ② Blocki verifies the local signature. If the output is 1, the data stored in CSPi is undamaged; otherwise, the data stored in the CSPi is damaged.

1) SIGNATURE GENERATION PHASE

Local signature generation $l\sigma \leftarrow Sig'(commit_i, X_i, R_i)$.

CSPi calculates $c_i = H_{sig}(X_i, R_i, commit_i)$, where X_i and R_i are generated in (3), and $commit_i$ is generated in (1), and obtain

$$ls_i = r_i + c_i x_i \ mod\ p. \tag{7}$$

CSPi generates the local signature $l\sigma = (R_i, ls_i)$ and sends it to blocki of the blockchain.

2) SIGNATURE VERIFICATION PHASE

Algorithm 3 shows the process of the signature verification phase of local verification, and the detailed steps are as follows:

Step 1: Commitment generation creation $commit' \leftarrow CommitCreate'(Enc(\hat{m}_i))$.

CSPi gets v_i from blocki, generates the commitment $commit'$ and uploads it to blocki:

$$lcommit = Enc(\hat{m}_i) * F + v_i * H \tag{8}$$

Step 2: Local signature verification $0/1 \leftarrow verification'(lcommit, L, l\sigma)$.

Blockchain stores a set of public keys $L = \{X_1, \ldots, X_n\}$, local signature $l\sigma = (R_i, ls_i)$, the commitment $lcommit$, blocki calculates $c_i' = H_{sig}(X_i, R_i, lcommit)$, verifies that $ls_i * G = R_i + c_i' X_i$. If the output is 1, the data stored in CSPi is correct; otherwise, the data stored in the CSPi is damaged.

Algorithm 3 Signature Verification Phase of Local Verification

Input: $Enc(\widehat{m_i}), L, \sigma' = (R_i, s_i')$
Output: 0/1
1: CSPi generates $lcommit = Enc(\hat{m}_i) * F + v_i * H$;
2: Blocki calculates $c_i' = H_{sig}(X_i, R_i, commit')$;
3: **If** $ls_i * G == R_i + c_i' X_i$ **then**
4: $flag' = 1$;
5: Else
6: $flag' = 0$;
7: **Return** $flag'$

VI. SECURITY ANALYSIS

As the assumption of the threat model, the blockchain can correctly store the data uploaded by different entities and ensure the accurate execution of the verification process. This section focuses on the security of the overall verification phase, since the security derivation of local verification is similar to that of aggregate validation.

Definition 1: Elliptic curve discrete logarithm problem (ECDLP): given an elliptic curve E defined over a finite field, a point P of order n on E, and a point Q that is a multiple of P, and one has to find the integer $x \in [0, n-1]$ such that $Q = lP$.

Theorem 1 This scheme can resist Rogue Key Attacks:

Proof of Theorem 1: Suppose that the honest CSPA owns (r_a, R_a), (x_a, X_a), and there exists a malicious CSPB owns (R_b, X_b) and tries to cover the signature of honest CSPA with its own signature to achieve the purpose of control the aggregated signature. The malicious CSPB tries to control the aggregated public key to its own public key X_b by forging R_f and public key X_f.

$$\begin{aligned} R_f &= R_b - R_a \\ X_f &= X_b - X_a \end{aligned}$$

Therefore, CSPA and CSPB calculate the following values:

$$\begin{aligned} a_a &= H_{agg}(X_a, X_f, X_a) \\ a_f &= H_{agg}(X_a, X_f, X_f) \\ X &= a_a X_a + a_f X_f \\ R &= R_a + R_f = R_b \\ c &= H_{sig}(R, X, m) \end{aligned}$$

Then, the malicious CSPB tries to build a signature:

$$s_b = r_b + ck_s$$

The malicious CSPB does not know the value of k_s, but it wants to export k_s using the data it already has. For this signature to be valid, it must be verified as $R + cX$.

$$\begin{aligned} s_b G &= R + cX \\ (r_b + ck_s)\, G &= R_b + c\left(a_a X_a + a_f X_f\right) \\ &= R_b + c\left(a_a X_a + a_f X_b - a_f X_a\right) \\ &= \left(r_b + ca_a x_a + ca_f x_b - ca_f x_a\right) G \\ ck_s &= ca_a x_a + ca_f x_b - ca_f x_a \end{aligned}$$

It is impossible for the malicious CSPB to crack ck_s without knowing the private key x_a of the CSPA, so the scheme can resist Rogue Key Attacks.

Theorem 2: Blockchain and CSPs cannot obtain the plaintext information, which ensures data privacy and security.

Proof of Theorem 2: Data privacy security is that the adversary cannot obtain any original information without the corresponding key. The data security of this scheme includes the data security of CSPs and the data security of blockchain.

The data in CSPs include encrypted data $E(m_i)$, commitment $commit_i$, aggregate commitment $Sumcommit$, signature s_i, public key X_i. ① The DO uploads the encrypted data $E(m_i)$ to the CSP. The security of the encrypted data is ensured by the encryption algorithm. ② The probability of extracting $E(m_i)$ from commitment $commit_i$, extracting $\sum_{i=1}^{n} Enc(m_i)$from aggregate commitment $Sumcommit$, and cracking the private key x_i from public key X_i is equal to the probability of solving ECDLP which can be ignored. ③ The probability of extracting aggregate commitment $Sumcommit$ from signature $s_i = r_i + ca_i x_i modp, c = H_{sig}(\tilde{X}, R, Sumcommit)$ is equal to the probability of finding the inverse of the hash function H_{sig} which can be ignored.

The data in the blockchain includes the aggregate signature s for overall verification and the local signature ls_i for local verification. The proof is the same as ③.

Definition 2: Suppose that the adversary F is the $(t, q_s, q_h, N, \epsilon)$ forger in the random prediction model for this scheme, the running time of F is t, and the adversary F performs a maximum of q_s signature queries with the honest CSP and a maximum of q_h queries with the random oracle, and wins the secure game with a probability of at least ϵ. The size of L in any signature query and forgery is at most N.

Theorem 3: Suppose there exists an adversary $F\,(t, q_s, q_h, N, \epsilon)$ for an aggregate signature σ with group parameters (G, p, g), where p is the k-bit length, and the hash functions $H_{com}, H_{agg}, H_{sig} : \{0,1\}^* \rightarrow \{0,1\}^l$ are random oracle, then there exists an algorithm C that can solve the ECDLP with (t', ϵ'), where $t' = 4t + 4N(Exp_G + 1) + O(N(q_h + q_s + 1))$, Exp_G is an exponential operation in G, and

$$\epsilon' \geq \frac{\varepsilon^4}{(q_h + q_s + 1)^3} - \frac{16(q_h + Nq_s)}{2^k} - \frac{16(q_h + Nq_s)^2 + 3}{2^l}$$

Proof of Theorem 3. Assume that the key pair (x^*, X^*) is generated by an honest CSP. The public key X^* is provided as input to the adversary F. The adversary F can perform

protocol execution with the honest CSP by providing the message *Sumcommit* and the multiple sets of public keys L involved in the signing process where X^* occurs at least once, and simulating all CSP except one instance of X^*. More concretely, the adversary will interact with the signature oracle, which works as follows: The adversary F sends a set of public keys L ($X^* \in L$) and the signature information *Sumcommit* to the signature oracle; The signature oracle resolves L as $\{X_1 = X^*, \ldots, X_n\}$, select$r_1 \leftarrow Z_p$, calculate $R_1 = g^{r_1}$, and send $t_1 = H_{com}(R_1)$ to adversary F; the adversary F returns $t_2, \ldots, t_n$; the signature oracle sends R_1 to adversary F, and the adversary returns $R_2, \ldots, R_n$; the signature oracle aborts the protocol if $t_i = H_{com}(R_i)$ ($i \in [2, \ldots, n]$), otherwise it computes

$$R = \sum_{i=1}^{n} R_i$$
$$c = H_{sig}(\tilde{X}, R, Sumcommit)$$
$$s_1 = r_1 + ca_1x_1 \ mod \ p$$

sends s_1 to adversary F; the adversary F sends $s_2, \ldots, s_n$ to the signature oracle; the signature oracle output $s = \sum_{i=1}^{n} s_i$.

The key to the proof is how to extract the discrete logarithm x^* from the public key X^*. The standard technique ''Forking Lemma'' [22] is to perform two ''fork'' in order to obtain two valid forgery tuples (s, R) and (s', R') for the same public key set L ($X^* \in L$) and the aggregation commitment *Sumcommit*. $R = R'$, $H_{sig}(\cdot)$ was programmed in both executions to the same value h_1, $H_{agg}(L, X_i)$ was programmed in both executions to the same value a_i for each i such that $X_i = X^*$, and $H_{agg}(L, X^*)$ was programmed to two distinct values $h_0$and h_0' in the two executions, implying that

$$s * G = R + (n^*h_1h_0)(X^*) + \sum_{i=1, X_i \neq X^*}^{n} (a_ih_1X_i)$$
$$s' * G = R + (n^*h_1h_0')(X^*) + \sum_{i=1, X_i \neq X^*}^{n} (a_ih_1X_i)$$

where n^* is the number of times X^* appears in L, so C can compute the discrete log of X^* as

$$x^* = (s - s')(h_0 - h_0')^{-1} \ mod \ p$$

Ignoring the time to compute the discrete logarithm, the running time of C is the running time of F, the time of maintaining the tables $T_{com}, T_{agg}, T_{sig}$ and the time required to answer the signature query. The sizes of $T_{com}, T_{agg}, T_{sig}$ are at most $q_h + Nq_s$, qN and q respectively, where $q = q_s + q_h + 1$, the time required to answer mainly depends on the time required to compute the aggregation key and the time to generate the aggregation commitment, which is at most $N(t_{exp} + 1)$, therefore $t' = 4t + 4N(Exp_G + 1) + O(N(q_h + q_s + 1))$.

The method of aggregation commitment signature in this scheme relies on Musig algorithm [21], and the success probability ϵ' of C is at least as

$$\epsilon' \geq \frac{\varepsilon^4}{(q_h + q_s + 1)^3} - \frac{16(q_h + Nq_s)}{2^k} - \frac{16(q_h + Nq_s)^2 + 3}{2^l}$$

VII. PERFORMANCE

A. NUMERICAL ANALYSIS

We assume Mul_G is multiplication operation on G, Add_G is addition operation on G, $Hash_G$ is the hash operation on G, $Pair_G$ is bilinear pair operation on G, Exp_G is the exponentiation operation on G, Mul_Z is the multiplication operation on Z_q, Add_Z is the addition operation on Z_q.

TABLE 2. The computational overhead of the scheme.

verification mode	signature generation phase	signature verification phase
overall verification	$6Add_G + 7Mul_G + 3T_H$	$2Add_Z + 4Mul_G + 2Hash_G$
local verification	$Add_G + Mul_G + Hash_G$	$2Add_Z + 4Mul_G + 2Hash_G$

There are two verification modes in this scheme: overall verification and local verification. Each mode consists of the signature generation phase and the signature verification phase. TABLE 2 shows the computational cost of the two modes in different stages of this scheme. For overall verification, the computational overhead in the signature generation phase represents the computational time of a CSP or a data owner, and the computational overhead in the signature verification phase represents the computational time of a CSP or a block of blockchain.

1) OVERALL VERIFICATION

The computational cost of signature generation phase includes commitment generation by DOs, the aggregate commitment generation and aggregate signature generation by CSPs. The computational cost of generating commitment is $2Mul_G + Add_G$, the computational cost of generating aggregated commitment is Add_G, and the computational cost of generating aggregated signature is $5Mul_G+4Add_Z+3Hash_G$, Therefore, the computational cost of signature generation phase is $6Add_G + 7Mul_G + 3Hash_G$.

The computational cost of signature verification phase includes of aggregate commitment generation by CSPs and signature verification by Blockchain. The computational cost of commitment generation is $2Mul_G + Add_G$, and the computational cost of signature verification is $Add_Z + 2Mul_G + 2Hash_G$, so the computational cost of signature verification phase is $2Add_Z + 4Mul_G + 2Hash_G$.

2) LOCAL VERIFICATION

In the signature generation phase, the required public key X_i, R_i and commitment $commit_i$ are already generated in the signature generation phase of the overall verification, so the computational cost of signature generation phase is $Add_G + Mul_G + Hash_G$.

The computational cost of signature verification phase includes CSP commitment generation by CSPs and local signatures verification by Blockchain. Similar to the overall

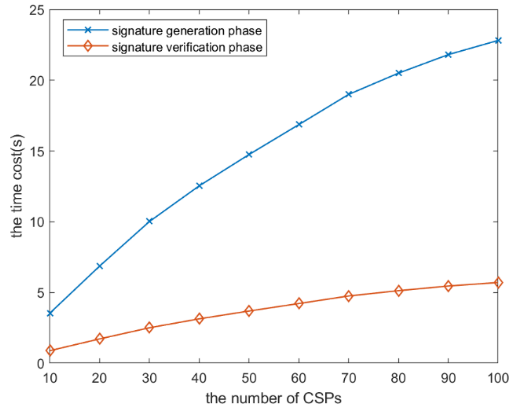

FIGURE 3. The time cost of overall verification.

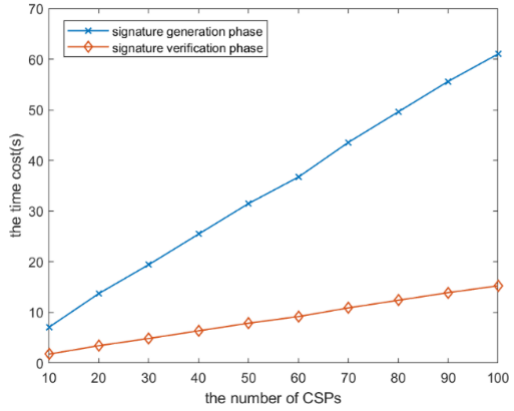

FIGURE 4. The time cost of local verification.

verification, the computational cost of signature verification phase is $2Add_Z + 4Mul_G + 2Hash_G$.

B. EXPERIMENTAL RESULTS

In this section, we evaluate the performance of the scheme. The scheme is executed on the test computer with the following configurations: CPU: Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz 2.11 GHz, RAM:8.0GB; System type: 64-bit OS. The secure hashing algorithms used in the experiment are SHA-256 and SHA-512. Assume that each data owner holds a file size of 30M.

To evaluate the performance of overall verification, this paper provides cloud storage services with10-100 CSPs for 10-100 DOs, and tests the time cost of signature generation phase and signature verification phase with the increase in the number of DOs, as shown in Figure 3. In Figure 3, the time cost of signature generation and signature verification shows a logarithmic increase and tends to be stable with the increase of the number of DOs. This is because DOs and CSPs can execute the algorithm *CommitGen*(), *Sign*(), *verification*() at the same time, which speeds up the execution of the algorithm.

To evaluate the performance of local verification, we test the local verification method in the cloud storage system with10-100 CSPs for 10-100 DOs, as shown in Figure 4. In Figure 4, the time cost of signature generation and signature verification increases linearly with the increase of the number of DOs, because the local verification adopts the method of one-by-one verification to ensure the accuracy of verification.

In addition, we compared the time cost of overall verification with the other blockchain-based multi-cloud storage data audit scheme [10], and the experimental results are shown in Figure 5. In Figure 5, when the number of CSP is small, the time cost of the scheme is slightly longer than that of the scheme [10], but with the increase of the number of CSP, the time cost of the scheme is significantly smaller than that of the scheme [10], indicating that this scheme can generate signatures quickly for a large number of CSPs.

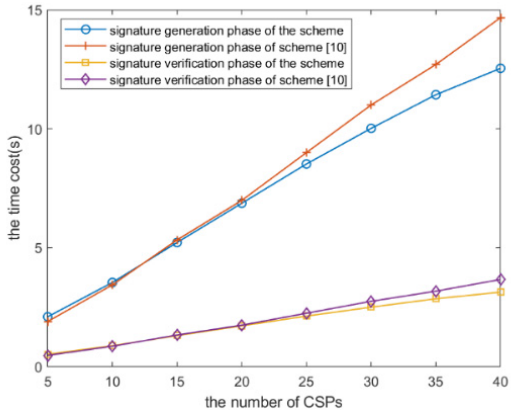

FIGURE 5. The compared time cost of overall verification.

VIII. CONCLUSION

This paper proposes a Blockchain-based efficient data integrity verification scheme in multi-cloud storage. We design two data integrity verification modes: overall verification and local verification. In the overall verification, we generate the aggregated commitment for the data in multiple CSPs with Pedersen commitment technology, and the Musig technology is employed to sign the aggregated commitment. This method can verify the data integrity of multiple CSPs and resist Rough Key attack. Local verification can find the specific CSP whose data integrity has been broken. In addition, the data verification process is directly arranged in the blockchain for public execution, providing data integrity verification services without the assistance of any third-party platform, thus avoiding security problems caused by untrusted TPA. In the future work, we will consider optimizing the scheme to resist replay attacks and increase the security of the scheme.

REFERENCES

[1] H. Li, F. Guo, and L. Wang, "A blockchain-based public auditing protocol with self-certified public keys for cloud data," *Secur. Commun. Netw.*, vol. 2021, Feb. 2021.
[2] D. Yaga, P. Mell, and N. Roby, "Blockchain technology overview," 2019, *arXiv:1906.11078*.

[3] R. Yang, F. R. Yu, P. Si, Z. Yang, and Y. Zhang, "Integrated blockchain and edge computing systems: A survey, some research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1508–1532, Feb. 2019.

[4] Y. Xu, C. Zhang, G. Wang, Z. Qin, and Q. Zeng, "A blockchain-enabled deduplicatable data auditing mechanism for network storage services," *IEEE Trans. Emerg. Topics Comput.*, vol. 9, no. 3, pp. 1421–1432, Jul. 2020.

[5] B. Liu, X. L. Yu, S. Chen, X. Xu, and L. Zhu, "Blockchain based data integrity service framework for IoT data," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jun. 2017, pp. 468–475.

[6] F. Xiang, W. Huaimin, S. Peichang, F. Yingwei, and W. Yijie, "JCLedger: A blockchain based distributed ledger for JointCloud computing," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. Workshops (ICDCSW)*, Jun. 2017, pp. 289–293.

[7] W. Shen, J. Qin, J. Yu, R. Hao, J. Hu, and J. Ma, "Data integrity auditing without private key storage for secure cloud storage," *IEEE Trans. Cloud Comput.*, vol. 9, no. 4, pp. 1408–1421, Oct. 2021.

[8] Y. Xu, J. Ren, Y. Zhang, C. Zhang, B. Shen, and Y. Zhang, "Blockchain empowered arbitrable data auditing scheme for network storage as a service," *IEEE Trans. Services Comput.*, vol. 13, no. 2, pp. 289–300, Mar. 2019.

[9] J. Xue, C. Xu, J. Zhao, and J. Ma, "Identity-based public auditing for cloud storage systems against malicious auditors via blockchain," *Sci. China Inf. Sci.*, vol. 62, no. 3, pp. 1–16, Mar. 2019.

[10] C. Zhang, Y. Xu, Y. Hu, J. Wu, J. Ren, and Y. Zhang, "A blockchain-based multi-cloud storage data auditing scheme to locate faults," *IEEE Trans. Cloud Comput.*, early access, Feb. 8, 2021, doi: 10.1109/TCC.2021.3057771.

[11] Y. Deswarte, J. J. Quisquater, and A. Saïdane, "Remote integrity checking," in *Proc. Working Conf. Integrity Internal Control Inf. Syst.*, 2003, pp. 1–11.

[12] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proc. 14th ACM Conf. Comput. Commun. Secur. (CCS)*, 2007, pp. 598–609.

[13] S. Shin and T. Kwon, "A survey of public provable data possession schemes with batch verification in cloud storage," *J. Internet Services Inf. Secur.*, vol. 5, no. 3, pp. 37–47, 2015.

[14] Q. Wang, C. Wang, and J. Li, "Enabling public verifiability and data dynamics for storage security in cloud computing," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2009, pp. 355–370.

[15] B. Wang, B. Li, and H. Li, "Oruta: Privacy-preserving public auditing for shared data in the cloud," *IEEE Trans. Cloud Comput.*, vol. 2, no. 1, pp. 43–56, Jan./Mar. 2014.

[16] Y. Yu, L. Xue, M. H. Au, W. Susilo, J. Ni, Y. Zhang, A. V. Vasilakos, and J. Shen, "Cloud data integrity checking with an identity-based auditing mechanism from RSA," *Future Gener. Comput. Syst.*, vol. 62, pp. 85–91, Sep. 2016.

[17] 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, Jul. 2019.

[18] Y. Zhang, R. H. Deng, X. Liu, and D. Zheng, "Blockchain based efficient and robust fair payment for outsourcing services in cloud computing," *Inf. Sci.*, vol. 462, pp. 262–277, Jun. 2018.

[19] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Proc. Annu. Int. Cryptol. Conf.*, 1991, pp. 129–140.

[20] A. Ibrahim and M. Singhal, "A new authentication protocol for an Authentication-as-a-Service (AaaS) cloud using Pedersen commitment scheme," in *Proc. Int. Conf. Ind. Informat. Comput. Syst. (CIICS)*, Mar. 2016, pp. 1–6.

[21] G. Maxwell, A. Poelstra, Y. Seurin, and P. Wuille, "Simple schnorr multi-signatures with applications to bitcoin," *Des., Codes Cryptogr.*, vol. 87, no. 9, pp. 2139–2164, Sep. 2019.

[22] D. Pointcheval and J. Stern, *Security Proofs for Signature Schemes* (Lecture Notes in Computer Science), vol. 1070. Berlin, Germany: Springer, 1996, pp. 387–398.

YIRAN ZHANG received the B.E. degree in computer science and technology from Zhengzhou University, China, in 2018, and the M.S. degree in computer application technology from Wuhan University, China, in 2021. She is currently a Researcher with the China Mobile Research Institute, Beijing, China. Her current research interests include data security and multimedia information security.

HUIZHENG GENG received the M.S. degree in information security from Shandong University, in 2014. He is currently a Technical Manager at the China Mobile Research Institute. He has eight years experience in the field of data security. His current research interests include data security and network security.

LI SU received the Ph.D. degree in information security from the Huazhong University of Science and Technology, in 2008. He is currently the Vice Director of the Security Technology Department, China Mobile Research Institute. He has 14 years experience in the field of information security of telecommunications and holds the certificate of Certified Information Systems Security Professional (CISSP). His current research interests include network security and information security.

LI LU received the M.S. degree in information security from Beijing Jiaotong University, in 2016. She is currently a Researcher at the China Mobile Research Institute. She has six years experience in the field of data security and network security. Her current research interests include data security and network security.