Fine-Grained Outsourced Data Deletion Scheme in Cloud Computing

Changsong Yang[1], Qiyu Chen[2], and Yueling Liu[1]
(Corresponding author: Changsong Yang)

School of Computer Science and Information Security, Guilin University of Electronic Technology[1]
No. 1 Jinji Road, Guilin, Guangxi, China
College of Information and Computer Engineering, Northeast Forestry University[2]
No. 26 HeXing Road, Harbin, Heilongjiang, China
(Email: csyang02@163.com)

(Received Aug. 23, 2019; Revised and Accepted Oct. 1, 2019; First Online Oct. 10, 2019)

Abstract

With the rapid development of cloud computing, more and more data owners embrace cloud storage service, by which they can outsource their data to the cloud server to greatly reduce the local storage overhead. However, in cloud storage, the outsourced data deletion becomes a severe secure challenge, because the cloud server might not honestly perform the data deletion for financial benefits. To solve this problem, we put forward a fine-grained data deletion scheme for cloud storage. In the proposed scheme, we introduce committed Rank-based Merkle hash tree chain, and adopt it to generate related evidences, which will be used to verify the storage and deletion results. Moreover, the proposal can achieve public verifiability without requiring any trusted third party, which exceeds plenty of previous solutions. Finally, the security analysis and simulation experiments demonstrate that the proposed scheme not only can satisfy the desired design goals, but also can achieve the efficiency and practicality.

Keywords: Cloud Storage; Data Deletion; Public Verifiability; Rank-based Merkle Hash Tree; Vector Commitment

1 Introduction

With the rapid development of information technology, the amount of the data increases continuously at an unprecedented speed. According to the investigation report of IDC [14], from 2013 to the end of 2020, the global data volume will grow exponentially and will reach 44 ZB from 4.4 ZB (about 1.8 trillion GB). The rapid increasing of the data brings a severe challenge: the data owner cannot maintain so many data locally. Fortunately, cloud storage provides an attractive solution to solve this problem [23]. By employing cloud storage service, the resource-constraint data owners, including the individuals and enterprises can outsource their large-scale data to the remote cloud server, which can greatly reduce the local storage overhead. Hence, more and more data owners are willing to embrace cloud storage service [25]. It's reported that 82% of the enterprises can save software/hardware investments by embracing cloud storage service [11].

Because of the promising market prospect, a large number of companies invest cloud storage, e.g., Amazon, Microsoft, Quidway, Alibaba, resulting in a various of cloud storage services. These cloud storage services have a big difference in price, security, access speed, reliability, etc. Meanwhile, an increasing number of data centers will emerge. According to the report of Cisco [10], the number of the hyperscale data centers will reach 628 by the end of 2021, up from 338 in 2016. As a result, the storage ability of the data centers will grow rapidly. Compared with 2016, the data center storage installed capacity will show a 4-fold increase by 2021 (about 2.6 ZB). Meanwhile, the data stored in data centers will reach 1.3 ZB by 2021. In other words, cloud storage service will attract considerable attentions.

Despite the tremendous benefits, cloud storage inevitably suffers from some new security challenges. Firstly, both the internal attackers (e.g., the administrator of the cloud) and the external attackers (e.g., hacker) may try to obtain some sensitive information from the outsourced data, which may cause privacy disclosure. Hence, the outsourced data may suffer from privacy disclosure in cloud storage. Secondly, the cloud server and the hacker might delete some data that are rarely accessed or insert some unrelated data, which will destroy the outsourced data integrity. Meanwhile, the data owner is very difficult to find these malicious behaviors. Last but not least, when the data owner will not need the outsourced data anymore, he will require the cloud server to delete the data. However, the selfish cloud server might not honestly execute deletion command for saving overhead or digging some implicit benefits. Therefore, the malicious data reservation is unexpected from the data owner's point of view. In short, although cloud storage service is economically attractive, it also inescapably suffers from some serious security challenges, specifically for the data confidentiality, data integrity, data deletion. If these severe challenges are not solidly solved, it will impede the public to accept and employ the cloud storage service.

1.1 Our Contribution

In this paper, we put forward a novel publicly verifiable and fine-grained data deletion scheme for cloud storage. In the proposed scheme, we consider the scenario that the cloud server deletes the outsourced data and returns a deletion evidence, which will be used to verify the deletion result by the data owner. Thus, the main contributions of this paper are as follows.

- We study the problems of the outsourced data secure storage and assured deletion, and then put forward a fine-grained data deletion scheme. The proposed scheme allows the data owner to efficiently check the outsourced data storage and deletion results by verifying the corresponding evidences. If the cloud server doesn't maintain/delete the data and generate the evidence honestly, the data owner can detect the malicious behaviors with a non-negligible probability.
- Inspired by the Rank-based Merkle Hash Tree (RMHT) and the vector commitment (VC), we introduce committed Rank-based Merkle Hash Tree Chain (CRMHTC). Then we adopt CRMHTC to achieve publicly verifiable data storage and deletion without requiring any trusted third party, which exceeds plenty of previous schemes. Moreover, the proposed scheme is very efficient in communication as well as computation.

1.2 Related Works

Secure data deletion has become a research hotspot, and has attracted plenty of attentions from academia and industry. The most traditional data deletion method is unlinking, which can efficiently remove the link of the file. However, the contents still remain in the storage medium. The adversary can apply tool to scan the disk to recover the deleted file [3]. Therefore, the researchers suggest that it should achieve deletion by overwriting [13, 15, 16] or by cryptography [5, 6, 21, 24].

To delete the contents of the file, Paul and Saxena [12] put forward a novel cloud data deletion scheme called proof of erasability (PoE). In their scheme, they reach data deletion by utilizing random patterns to overwrite the disk. After that the storage medium returns the same patterns as a deletion evidence. However, the pattern of the data will be revealed if some one else (not the data owner) wants to verify the deletion result. In 2016, Luo *et al.* [9] presented a new scheme to achieve outsourced data deletion by overwriting. They assume that the cloud server is so economical selfish that it merely stores the latest version of the data, and all the copies of the outsourced data will be consistent when they are updated. Therefore, they realize data deletion by updating the file with random data. Finally, the data owner can check the deletion result through a challenge-respond protocol. However, the above two schemes [9, 12] are inefficient because the overwriting operation might cost a lot of computational overhead. Xiong *et al.* [17] presented a secure cloud data self-destructing scheme. In their scheme, the keys are associated to a time interval. If the time exceeds the time interval, the keys will be destroyed automatically. Although their scheme can reach efficient cloud data deletion, it must artificially set up the time interval in advanced.

Du *et al.* [2] designed an address-based multi-copy associated deletion protocol in 2018. In their scheme, they use physical and logical addresses to locate a copy uniquely. Moreover, they utilize deleting sequence and Merkle hash tree to achieve data integrity verification and verifiable deletion. However, their scheme needs to introduce a third party to manage the data keys. Xue *et al.* [19] put forward a key-policy attribute-based encryption scheme, which reaches data deletion by removing the related attribute. Then they use Merkle hash tree to achieve verifiability of the deletion result. Moreover, their scheme can realize fine-grained access control, but it also requires a trust authority. To overcome the bottleneck of requiring a trusted third party, Yang *et al.* [20] proposed a blockchain-based publicly verifiable data deletion scheme for cloud storage. Their scheme makes use of the advantages of blockchain to achieve public verifiability without requiring any trusted third party.

To simultaneously realize secure outsourced data transfer and deletion, Xue *et al.* [18] put forward a novel scheme, which is characterized by provable data possession, provable data transfer and verifiable data deletion. In their scheme, the data owner can migrate the data from one cloud server to another one, and check the data integrity through provable data possession scheme. Besides, their scheme achieves assured deletion by using the primitive of Rank-based Merkle hash tree. However, Liu *et al.* [8] pointed out that there is a security flaw in scheme [18]: the dishonest cloud can arbitrarily modify a data block and forge a corresponding block tag that can pass verification. After that they designed an improved data transfer and deletion scheme which can resist the above attack. Moreover, compared with scheme [18], their scheme is more efficient in data integrity checking. However, both of the above two schemes need a third party auditor. To solve this problem, Yang *et al.* [22] constructed a vector commitment-based publicly verifiable cloud data transfer and deletion scheme. Their scheme can provide the data owner with the ability to migrate the outsourced data to another cloud, and then delete the transferred data from the original cloud securely. Finally, the data owner can verify the transfer and deletion results without requiring any third party.

1.3 Organization

The rest of this paper are organized as follows: The problem statement is described in Section 2. In Section 3, we present the preliminaries, including the Rank-based Merkle Hash Tree, the bilinear pairings, the computational Diffie-Hellman problem and the vector commitment. The committed Rank-based Merkle Hash Tree Chain and the detailed scheme are presented in Section 4. Meanwhile, we analyze the security in Section 5, and provide the efficiency evaluation through simulation experiments in Section 6. Finally, we briefly conclude this paper in Section 7.

2 Problem Statement

In this section, we will present the problem statement in detail, including the system and threat model, and the main design goals.

2.1 System Model

In cloud storage, the resource-constraint data owner outsources his large-scale personal data to the remote cloud server, which can greatly reduce the local storage overhead. Therefore, the system model of our scheme contains two entities: the local data owner and the remote cloud server, as shown in Figure 1.

- **The data owner** is an entity that owns limited network bandwidths, storage resources and computation resources. Hence, the data owner prefers to outsource his large-scale personal data to the remote cloud server for saving the local storage overhead. Besides, the data owner wants to verify the data storage/deletion results to ensure that the cloud server maintains/deletes the data honestly.

- **The cloud server** refers to an entity that has plentiful storage resources, and it utilizes these resources to provide data owners with high-quality cloud storage service. When the data owner will not need the data anymore, the cloud server executes deletion command to remove the data and generates a related evidence, which will be used to verify the deletion result by the data owner.

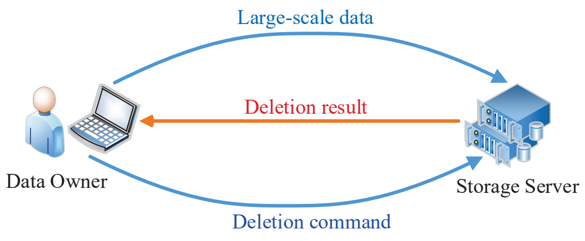

Figure 1: The system model of our scheme

2.2 Security Challenges

In cloud storage, there is a trust problem between the data owner and the cloud server. That is, the data owner doesn't trust that the selfish cloud server will maintain/delete the outsourced data honestly. Additionally, the attackers may try their best to access the outsourced data for digging some sensitive information. Last but not least, the outsourced data may be destroyed for some controllable or uncontrollable factors. Therefore, we should consider the following three security challenges.

- **Privacy disclosure.** From the data owner's point of view, the sensitive information should be prevent from disclosure. However, both the internal attacker (e.g., the administrator of the cloud) and the external attacker (e.g., the hacker) are so curious that they try their best to dig the privacy information from the outsourced data. Therefore, the privacy disclosure has become a severe security challenge in cloud storage.

- **Malicious data pollution.** In cloud storage, both the selfish cloud server and the attacker may destroy the outsourced data. On the one hand, the cloud server may arbitrarily delete some data that are rarely accessed for saving storage overhead. On the other hand, the attacker is so malicious that he may destroy the data designedly. Moreover, the software failure or hardware damage may cause the loss of the outsourced data.

- **Dishonest data reservation.** When the data owner no longer needs the outsourced data, he may require the cloud server to delete the data. However, the selfish cloud server may reserve the data maliciously for the following two reasons. Firstly, the deletion operation may cost some computational overhead, which is unexpected from the cloud server's point of view. Secondly, the cloud server can obtain some implicit benefits from the data reservation.

2.3 Design Goals

The new proposal should achieve the following three design goals.

- **Data confidentiality.** The sensitive information that contained in the outsourced data should be kept secret. Therefore, the local data owner must adopt secure cryptography algorithm to encrypt the outsourced file, and upload the corresponding ciphertext to the remote cloud server. Meanwhile, the data owner should maintain the related decryption keys secretly.

- **Data integrity.** The data integrity means that all the outsourced data blocks should be prevented from pollution. If the selfish remote cloud server and the adversary maliciously destroy the outsourced data blocks, e.g., deleting some data that are rarely accessed, inserting some unrelated data, these malicious behaviors will be detected by the data owner with a non-negligible probability.

- **Verifiable data deletion.** To guarantee that the cloud server deletes the related data blocks honestly, the remote cloud server should generate a corresponding data deletion evidence and return it to the local data owner. If the cloud server doesn't execute the data deletion command sincerely, it cannot generate an effective evidence to persuade the data owner that the data blocks have been deleted honestly.

3 Preliminaries

In this section, we may describe some preliminaries that will be utilized in our scheme later, including the Rank-based Merkle hash tree, the bilinear pairings, the computational Diffie-Hellman assumption and the vector commitment.

3.1 Bilinear Pairings

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic multiplicative groups of prime order p, and g be a generator of $\mathbb{G}_1$. The map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ is a bilinear pairing if it satisfies the following three properties:

- $Bilinear$. For all elements $X, Y \in \mathbb{G}_1$, and $a, b \in \mathbb{Z}_p^*$, $e(X^a, Y^b) = e(X, Y)^{ab}$.

- $Non-degenerate$. The equation $e(g, g) \neq 1$ always holds.

- $Computable$. For all elements $X, Y \in \mathbb{G}_1$, there always exists an algorithm that can efficiently compute the value of $e(X, Y)$.

CDH Problem. The computational Diffie-Hellman (CDH) problem can be defined as follows.

Definition 1. *The integers a, b are randomly chosen from $\mathbb{Z}_p$. On input g, g^a and g^b, then output g^{ab}.*

For all of the security parameter k, if for each probabilistic polynomial time (PPT) algorithm $\mathcal{A}$, there always exists such a negligible function $negl(\cdot)$ that $Pr[\mathcal{A}(1^k, g, g^x, g^y) = g^{xy} \leq negl(k)]$, then the CDH assumption holds.

3.2 Vector Commitment

Commitment, a particularly important cryptography primitive, which is applied in plenty of security protocols. Vector commitment (VC), an variant of the commitment, was first proposed by Catalano and Fiore in 2013 [1]. Generally speaking, a vector commitment scheme allows the sender to commit to an ordered sequence of message $(m_1, \cdots, m_q)$. After that the committer can open the commitment at specific positions. Meanwhile, no body is able to open the commitment to two or more different messages at a same position. This character is called *position binding*. Moreover, for a given vector commitment, although the adversary has known some opening values at corresponding positions, he cannot distinguish the given vector commitment is committed to the message sequence m or to the message sequence m', which is called *hiding*.

In the following, we will describe a CDH-based vector commitment scheme, which contains six algorithms.

- $VC.KenGen(1^k, q)$. Assume that $\mathbb{G}_1$ and $\mathbb{G}_2$ are two bilinear groups, whose orders both are the primer p. Besides, they satisfy the bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. Let $g \in \mathbb{G}_1$ be a random generator. $z_1, z_2, \cdots, z_q$ are randomly chosen from the message space $\mathbb{Z}_p$. Compute $h_i = g^{z_i}$, where $i = 1, 2, \cdots, q$. Meanwhile, set $h_{i,j} = g^{z_i z_j}$, where $i, j = 1, 2, \cdots, q$, $i \neq j$. Finally, the public parameters are $pp = (g, \{h_i\}_{i\in[1,q]}, \{h_{i,j}\}_{i,j\in[1,q],i\neq j})$

- $VC.Com_{pp}(m_1, \cdots, m_q)$. Compute the commitment value $C = h_1^{m_1} h_2^{m_2} \cdots h_q^{m_q}$, then output the commitment C and the auxiliary information $aux = (m_1, \cdots, m_q)$.

- $Open_{pp}(m_i, i, aux)$. Compute the evidence $\lambda_i = \prod_{j=1,j\neq i}^{q} h_{i,j}^{m_i} = (\prod_{j=1,j\neq i}^{q} h_j^{m_j})^{z_i}$.

- $VC.Ver_{pp}(C, m_i, i, \lambda_i)$. Check that whether the equation $e(C/h_i^{m_i}, h_i) = e(\lambda_i, g)$ holds. If not, output 0; otherwise, output 1.

- $VC.Update_{pp}(C, m, m', i)$. Compute and output an updated vector commitment $C' = C \cdot h_i^{m'-m}$. Meanwhile, output updated information $U = (m, m', i)$.

- $VC.ProofUpdate_{pp}(C, \lambda_i, m', U)$. This algorithm aims to compute updated commitment C' and a new proof λ'_j. We distinguish the following two cases:

  1) $i = j$. Compute the updated commitment C' as $C' = C \cdot h_i^{m'-m}$ while the updated proof remains the same λ_j.

  2) $i \neq j$. Compute the updated commitment C' as $C' = C \cdot h_i^{m'-m}$ and the updated proof is $\lambda'_j = \lambda_j \cdot (h_i^{m'-m})^{z_j} = \lambda_j \cdot h_{j,i}^{m'-m}$.

3.3 Rank-based Merkle Hash Tree

As a classical authentication data structure, Merkle Hash Tree (MHT) is always used to make sure that the data blocks are not maliciously destroyed and altered during storage and transmission [4, 7]. Rank-based Merkle Hash Tree (RMHT) is extended from the traditional MHT, which is very similar to the MHT. In a RMHT, the inputs of the hash function in a node are the hash value of the concatenation of its two child nodes and its rank, which is different from MHT. Here the rank is the number of leaf nodes this node contains. As shown in Figure 2, in the leaf nodes, the hash values $h_i = H(1||d_i)$, where $i = 1, 2, 3, 4$. In the internal nodes, $h_a = H(2||h_1||h_2)$, $h_b = H(2||h_3||h_4)$, and the root node $h_r = H(4||h_a||h_b)$. Similarly, a leaf node's verification object ϕ is a set of the hash values on the sibling path from the authenticated leaf node to the root node, such as the verification object of d_3 is $\phi = (h_4, h_a)$.

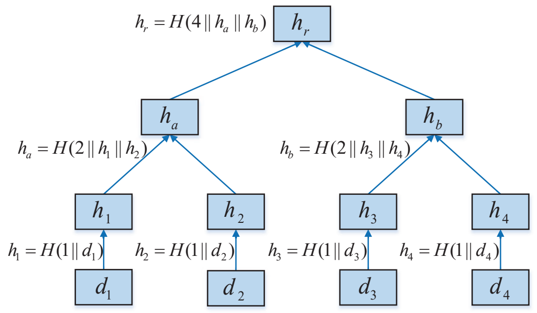

Figure 2: An Example of RMHT

4 Our Scheme

In the following section, we may put forward our verifiable and fine-grained outsourced data deletion scheme in detail. To be specific, we will firstly introduce the committed rank-based Merkle hash tree chain (CRMHTC), then give an overview of our scheme, and finally describe the detail construction.

4.1 The construction of CRMHTC

In this subsection, we introduce the committed rank-based Merkle hash tree chain (CRMHTC), which can be used to authenticate a large number of data blocks. A CRMHTC can be viewed as a combination of the vector commitment and the rank-based Merkle hash tree, as shown in Figure 3. Generally speaking, the CRMHTC contains q ordered rank-based Merkle hash trees, whose leaf nodes are used to maintain the data blocks. For simplicity, we assume that all the rank-based Merkle hash trees have the same height. That is, all the trees can store the same number of data blocks. Finally, a CDH-based vector commitment scheme is used to compute a commitment, which is created to all the root nodes of the rank-based Merkle hash trees.

4.2 Overview

In our scenario, the local data owner doesn't fully trust the remote cloud server, which is very similar to the previous schemes [6, 20]. More specifically, the data owner thinks that the cloud server may not store

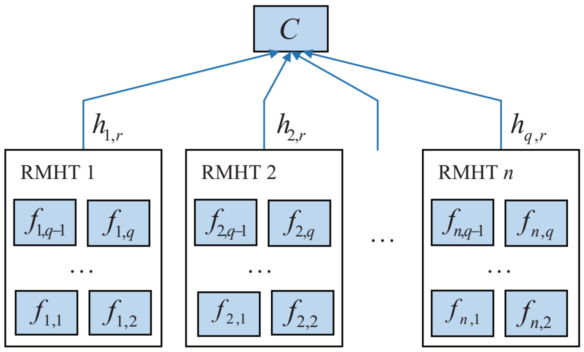

Figure 3: The construction of CRMHTC

or delete the data honestly. Therefore, the data owner wants to check the storage and deletion results to ensure that the cloud server behaves honestly. Our proposed scheme uses CRMHTC to achieve verifiable storage and deletion without requiring any trusted third party.

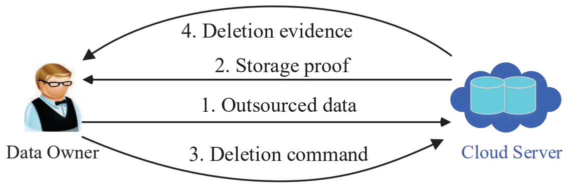

Figure 4: The main processes of our scheme

The main processes of our scheme are shown in Figure 4. Firstly, the data owner divides the outsourced file into q blocks and encrypts them with secure symmetric encryption algorithm, such as AES. After that the data owner further divides each block into n' sectors, and then inserts $n - n'$ random sentinels into every block at random positions, then records these random positions in a table PF. Whereafter, the data owner outsources the data to the remote cloud server. Upon receiving the data, the cloud server constructs a CRMHTC to maintain the received outsourced data, and returns a corresponding storage proof to the data owner. The data owner verifies the storage result and deletes the local backups. Finally, when the data owner will not need the data anymore, a deletion command is sent to the cloud server to delete the data. The cloud server executes deletion command and returns a deletion evidence, thus, the data owner can check the deletion result by verifying the deletion evidence.

4.3 The Concrete Construction

In the following, we will introduce our proposed scheme in detail. Note that the cloud service provider must authenticate the identity of the data owner before providing him with data storage service. For simplicity, we assume that the data owner has already passed the verification and become a legal user. Additionally, we assume that the data owner chooses a unique name n_f for the large-scale file F, which will be uploaded to the cloud server. Finally, the data owner random chooses two secure one-way collision resistant hash functions $H_1(\cdot)$ and $H_2(\cdot)$.

- **Initialization.** This phase firstly generates the ECDSA public private key pairs (pk_o, sk_o), (pk_s, sk_s) for the local data owner and the remote cloud server, respectively. Moreover, the cloud server runs the key generation algorithm $VC.KeyGen(1^k, q)$ to generate and publish the public parameters pp, where $pp = (g, \{h_i\}_{i\in[q]}, \{h_{i,j}\}_{i,j\in[q],i\neq j})$.

- **Data outsourcing.** The data owner encrypts his personal file to protect the data confidentiality, and then outsources the corresponding ciphertext to the cloud server.

  1) Firstly, the data owner divides the file F into q blocks $m_1, \cdots, m_q$. Then the data owner computes the data encryption keys $k_i = H_1(i||n_f||sk_o)$ and uses k_i to encrypt m_i as $f_i = Enc_{k_i}(m_i)$, where $1 \leq i \leq q$ and Enc is an IND-CPA secure symmetric encryption algorithm.
  2) Then the data owner further divides every block into n' sectors, and inserts $n - n'$ random sentinels into every block at random positions, and records these random positions in a table PF. Therefore, the outsourced data set D can be denoted as $D = \{d_{ij}\}_{1\leq i\leq q, 1\leq j\leq n}$. Finally, the data owner outources the data set D to the cloud server.

- **Data storage.** The cloud server maintains the data set D and returns a corresponding storage proof to the data owner.

  1) Upon receiving D, the cloud server maintains every data block in a RMHT, as shown in Figure 3. Then the cloud server defines the aux information as $aux = (R_1, R_2, \cdots, R_q)$, where R_i is the root of the RMHT in position i. Meanwhile, the cloud server computes signatures $S = (sig_1, sig_2, \cdots, sig_q)$ as $sig_i = Sign_{sk_s}(R_i)$, where $Sign$ is the ECDSA signature generation algorithm.
  2) After that the remote cloud server will use the information aux to run the committing algorithm $VC.Com_{pp}(R_1, \cdots, R_q)$ to compute a vector commitment C. Finally, the cloud server returns the storage proof $\omega = (C, S, aux)$ to the data owner.

- **Storage check.** The data owner checks the storage result by verifying the returned storage proof, and finally deletes the local backup to save storage overhead greatly.

  1) On receipt of proof ω, the data owner firstly checks the correctness of the roots. To be specific, for every RHMT, the data owner randomly chooses a leaf node and the corresponding verification object to compute a new root and compares it with R_i. If it is not true, the data owner quits and outputs FALSE; otherwise, the data owner goes to Step 2.
  2) The data owner verifies the validity of the signatures $sig_1, sig_2, \cdots, sig_q$. If not all of the signatures are valid, the data owner outputs FALSE and requires the cloud server to anew generate the valid storage proof; otherwise, the data owner runs $VC.Com_{pp}(R_1, \cdots, R_q)$ to obtain a new commitment and compares it with C. If it is not true, the data owner quites and outputs FALSE; otherwise, the data owner trusts that the storage proof is valid, and deletes the local data backup

- **Data deletion.** When the data owner will not need the data anymore, he might require the cloud server to delete the data and return a related data deletion evidence.

  1) Without lose of generality, we assume that the data owner wants to delete the sector $d_{i,j}$. First of all, the data owner generates a signature $sig_d = Sign_{sk_o}(delete||n_f||i||j||T_d)$, where T_d is a timestamp. After that the data owner generates a data deletion request $R_d = (delete, n_f, i, j, T_d, sig_d)$ and sends it to the cloud server.

2) Upon receipt of the deletion request R_d, the cloud server verifies the validity of R_d. If R_d is not valid, the cloud server quits and outputs FALSE; otherwise, the cloud server sends the data $d_{i,j}$ and the corresponding verification object $\phi_{i,j}$ to the data owner. Then the cloud server deletes $d_{i,j}$ and computes a new root R'_i (Figure 5 shows an example of deleting a data block from the RMHT).

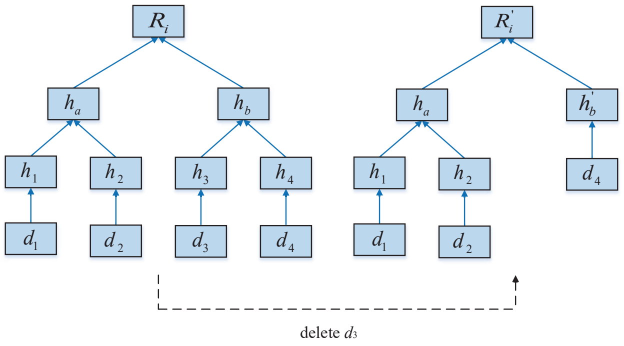

Figure 5: The RMHT updating for data deletion

3) After that the cloud server runs the update algorithm $VC.Update_{pp}(C, R_i, R'_i, i)$ to obtain a new commitment C' and update information $U = (R_i, R'_i, i)$. Meanwhile, the cloud server computes a signature $sig'_i = Sign_{sk_s}(R'_i)$. Finally, the cloud server returns the deletion evidence $\tau = (C', U, sig'_i)$ to the data owner.

- **Deletion check.** After receiving the data deletion evidence $\tau = (C', U, sig'_i)$, the data owner wants to check the deletion result.

1) Firstly, the data owner uses data $d_{i,j}$ and verification object $\phi_{i,j}$ to compute a new root node $\overline{R}_i$, and then compares it with R_i. If $\overline{R}_i \neq R_i$, the data owner quites and outputs FALSE; otherwise, the data owner verifies the validity of the signature sig_i. If it is not true, the data owner quites and outputs FALSE; otherwise, the data owner goes to Step 2.

2) The data owner utilizes $\phi_{i,j}$ to compute a new root R^*_i of the RMHT which has deleted $d_{i,j}$, and then compares it with R'_i. If $R^*_i \neq R'_i$, the data owner quites and outputs FALSE; otherwise, the data owner checks that whether the signature sig'_i is valid. If it is not true, the data owner quites and outputs FALSE; otherwise, the data owner goes to Step 3.

3) The data owner runs the proof update algorithm to obtain a new commitment C^* and checks that whether the equation $C^* = C'$ holds. If and only if $C^* = C'$, the data owner trusts that the deletion evidence is correct. If the data $d_{i,j}$ discovered on the cloud server later, the data owner can be entitled to compensation.

5 Security Analysis

5.1 Data Confidentiality

In our scenario, the data confidentiality means that the adversary cannot acquire any plaintext information from the outsourced ciphertext without the data decryption key. In the proposed scheme, the data owner utilizes the IND-CPA secure AES algorithm to encrypt the outsourced file before uploading. Note that the data decryption keys are computed by the data owner as $k_i = H_1(i||n_f||sk_o)$, where $1 \leq i \leq q$, sk_o is the private key and $H_1(\cdot)$ is a secure one-way collision resistant hash function. Hence, if the adversary doesn't have the private key, he cannot forge a valid data decryption key with a non-negligible probability. Moreover, the data owner keeps the private key sk_o and the data decryption keys $\{k_i\}_{i\in[1,q']}$ secret. Therefore, the adversary cannot obtain the data decryption keys to further acquire some plaintext information. That is, the proposed scheme can guarantee the outsourced data confidentiality.

5.2 Data Integrity

The outsourced data integrity means that if the outsourced data blocks are destroyed arbitrarily, the data owner can detect the malicious data pollution with a non-negligible probability. In the proposed scheme, the local data owner outsources the large-scale data to the remote cloud server to save the local storage overhead. Then the cloud server maintains the outsourced data blocks in the CRMHTC, and returns the storage proof $\omega = (C, S, \lambda, aux)$. The data owner executes the verification algorithm $VC.Ver_{pp}(C, R_i, i, \lambda_i)$ and checks the output. If and only if the output is one will the data owner further utilize the data block $d_{i,j}$ and the verification object to re-compute a new root r'_i and compares it with the hash value R_i. Note that $H_2(\cdot)$ is a secure one-way collision resistant hash function. Hence, the cloud server cannot forge a new data $d'_{i,j}$ to make the equation $r'_i = R_i$ hold with a non-negligible probability. That is, if the cloud server does not maintains data block $d_{i,j}$ honestly or the data block $d_{i,j}$ has already been destroyed, the data owner can detect these malicious behaviors with a non-negligible probability. Further, our scheme can achieve the outsourced data integrity.

5.3 Verifiable Data Deletion

To guarantee that the RMHT will not be empty, the data owner inserts some random sentinels into the outsourced data at random positions before uploading. When the data owner will not needs some data blocks, or even the whole file, he will require the cloud server to delete these data blocks. However, the inserted random sentinels still remain in the RMHT to generate the deletion evidence. The data owner can utilize the verification object to compute a new root value R^*_i and then compares it with R'_i that received from the cloud server. The data owner might have deleted some blocks before, which may change the path from one block to the root, even during this data deletion process the path is also changing. That is, there are many paths which can reach the root from the remaining nodes. We assume that there are l blocks left in the RMHT, there are $\frac{(2l-2)!}{l!(l-1)}$ ways to generate the root of the RMHT. Therefore, the cloud server cannot guess the right path with a non-negligible probability if it doesn't execute the data deletion command honestly. Further, the cloud server almost cannot generate the correct root R'_i. Therefore, if the root passes the verification, the cloud server has indeed deleted the data block. Moreover, the verification process does not require any private information. That is, the proposed scheme can achieve publicly verifiable data deletion.

6 Efficiency Analysis

In this section, we will firstly analyze the computational overhead in theory and then provide the efficiency evaluation through simulation experiments.

6.1 Comparison

In this part, we firstly compare the main theoretical functionality among schemes [6, 19] and our new proposed scheme. Then we analyze the efficiency from the point of the data owner and the cloud server in theory.

The results of the functionality comparison are shown in Table 1 and we can have the following findings. Firstly, all these three schemes not only can protect the data confidentiality, but also can realize verifiable data deletion. Secondly, only our new proposed scheme is able to provide the local data owner with the ability to verify the outsourced data storage result and the data integrity, which can ensure that the remote cloud server maintains the outsourced data honestly. Moreover, our novel scheme doesn't contain any trusted third party, which exceeds the other two schemes, because the trusted third party has become the bottleneck in the verifiable deletion system.

Table 1: The functionality comparison among the three schemes

	Scheme [6]	Scheme [19]	Our scheme
Trusted third party	Yes	Yes	No
Verifiable deletion	Yes	Yes	Yes
Data confidentiality	Yes	Yes	Yes
Storage checking	No	No	Yes
Data integrity	No	No	Yes

Then we analyze the computational complexity in theory and the results are shown in Table 2. Here, we firstly introduce some symbols that will be utilized later. We denote by $\mathcal{E}$ a symmetrical encryption operation, $\mathcal{H}$ a hash computation, E an exponentiation computation. Additionally, the symbols $\mathcal{S}$ and $\mathcal{V}$ represent a signature generation and a signature verification operation, respectively. Then q is the size of the vector commitment, l is the height of the RMHT. Therefore, the total number of the outsourced data sectors is $q2^{(l-1)}$. For simplicity, we omit other operations such as the multiplication calculations and the communication overhead.

Table 2: The computational complexity comparison

	Data owner	Cloud server
Data outsourcing	$q\mathcal{H} + 1\mathcal{E}$	-
Data storage	-	$q\mathcal{S} + qE + (2^l - 1)q\mathcal{H}$
Storage check	$q\mathcal{V} + ql\mathcal{H} + qE$	-
Data deletion	$1\mathcal{S}$	$1\mathcal{S} + 1\mathcal{V} + l\mathcal{H} + 1E$
Deletion check	$2l\mathcal{H} + 2\mathcal{S} + 1E$	-

From Table 2 we can find that the data oursorcing, storage result verification and deletion result verification are executed by the data owner. Note that the computational overhead of these operations almost linearly increases. Additionally, these operations are one-time, and they can be accomplished off-line. Moreover, plenty of the expensive computations are executed by the cloud server in the data storage and data deletion processes. Hence, our scheme is efficient from the data owner's point of view.

6.2 Performance Evaluation

In the following, we provide the efficiency evaluation through the simulation experiments. We simulate the proposed scheme on a laptop which equipped with Linux operation system (Ubuntu 14.04), 8 GB main memory and Intel(R) Core(TM) i5-6200U processors running at 2.4 GHz. Additionally, we use the PBC library and the OpenSSL library to implement the related cryptography calculations. For simplicity, we ignore the multiplication computation and the communication overhead.

In the initialization phase, the proposed scheme requires some one-time computational overhead to prepare some keys and the public parameters. To be more specific, the proposed scheme firstly needs to generate two ECDSA public private key pairs. Then it needs to execute $q(q+1)/2$ modular exponentiation calculations to generate the public parameter pp, and Figure 6 shows the time cost of the initialization process. We can find that the time cost presents exponential growth, however, all the computations can be executed off-line. Hence, it will not affect the efficiency greatly.

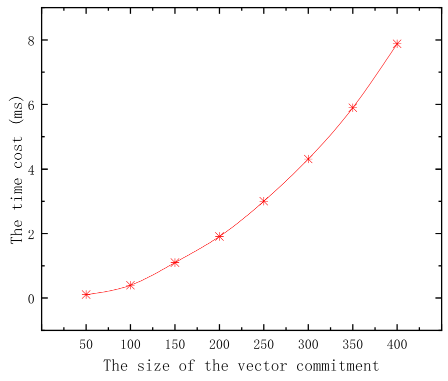

Figure 6: Time cost of initialization

The outsourced file always contains some sensitive information which should be kept secret from the data owner's point of view. That is, the data owner must use secure algorithm to encrypt the outsourced file before uploading. Therefore, in the data outsourcing process, the main computational overhead comes from the data encryption key generation and the data encryption. We increase the size of the encrypted file from 1 MB to 8 MB with a step for 1 MB. For simplicity, we fix the size of the vector commitment in 800 in this experiment, that is, $q = 800$, and test the approximate time cost, as shown in Figure 7. From Figure 7 we can easily find that the time cost will increase with the size of the encrypted data. However, the data encryption is one-time, hence, we think that our scheme is still efficient.

After receiving the outsourced data from the data owner, the cloud server maintains the data. Meanwhile, it generates a storage proof for the data owner. The main computations are hash calculations in constructing the rank-based Merkle hash trees, and the exponentiation computations in computing the value of the vector commitment. For simplicity, we assume that the height of every RMHT is the same. Hence, we increase the height of the RMHT from 1 to 8, and increase the size of the vector commitment from 50 to 400, and then Figure 8 shows the approximate computational overhead. From Figure 8 we can easily find that the time cost almost increases linearly with the size of the vector commitment (i.e., the number of the outsourced data blocks). Meanwhile, the time cost approximately increases exponentially with the height of the RMHT. However, this process is executed by the cloud server, which owns powerful computing ability. Moreover, the storage evidence generation operation is one-time, and it can be finished off-line. Therefore, it will not greatly affect the overall efficiency of the proposed scheme.

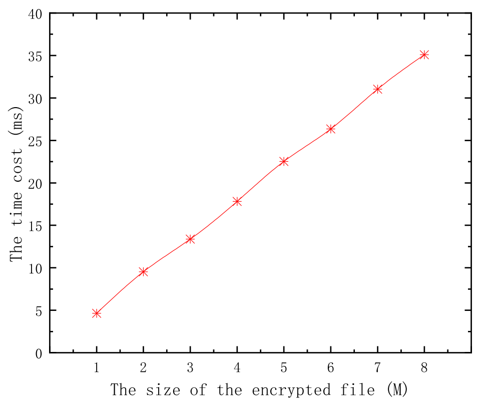

Figure 7: Time cost of data outsourcing

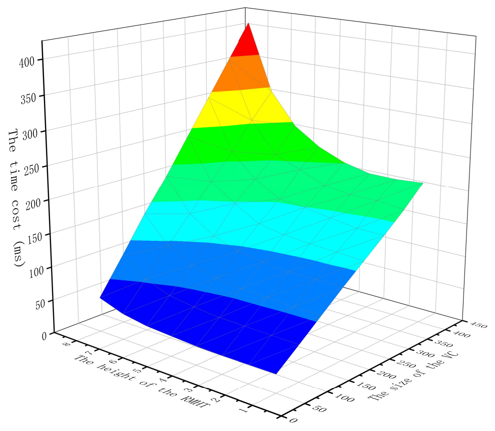

Figure 8: Time cost of data storage

Upon receiving the data storage evidence, the data owner can check the data storage result through verifying the storage evidence. To be specific, for every RMHT, the data owner re-computes the root node and compares it with the one received from the cloud server, and then verifies that whether the signatures are valid. Finally, the data owner checks the correctness of the vector commitment. Hence, we increase the height of the RMHT from 1 to 8, and increase the size of the vector commitment from 50 to 400, and the time cost is shown in Figure 9. We can find that the time overhead almost increases linearly with the size of the vector commitment and the height of RMHT. However, the growth rates are not high. Moreover, note that the time cost is not large, for example, when q reaches 350 and l reaches 8, the time cost is only 200 milliseconds.

When the local data owner will not need the outsourced data anymore, he signs a data deletion

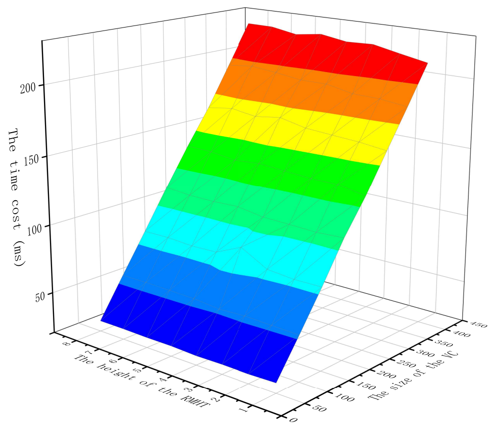

Figure 9: Time cost of data storage result verification

command to require the remote cloud server to delete the unnecessary data. Then the cloud server executes the data deletion command to delete the corresponding data and returns a data deletion evidence to the data owner. For simplicity, we assume that the data owner wants to delete a sector from the outsourced data set, and increase the height of the RMHT from 10 to 80 with a step for 10. Then we measure the approximate time overhead through the simulation experiments, as demonstrated in Figure 10. From Figure 10 we can easily find that the time cost of the data owner is almost constant. However, the time overhead of the cloud server increases with the height of the RMHT. Meanwhile, the time cost of the cloud server is larger than that of the data owner. That is, we can think that the cloud server undertakes most of the computational overhead. Therefore, from the data owner's point of view, it is quite efficient to delete the outsourced data.

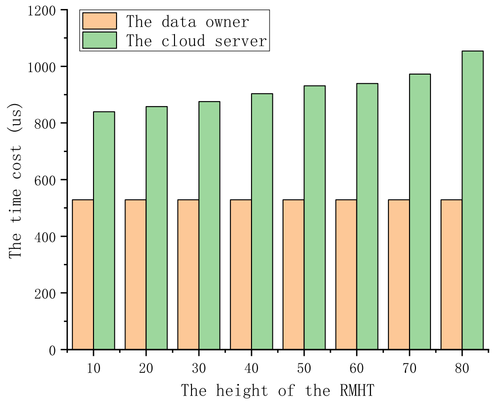

Figure 10: Time cost of data deletion

Finally, the data owner can check the data deletion result through verifying the returned data deletion evidence. To be more specifical, the data owner needs to generate two ECDSA signatures, execute a modular exponentiation calculation and compute $2l$ hash values. Similarly, we increase the height of the RMHT from 10 to 80 with a step for 10 in this experiment. Then we test the time cost and use it to evaluate the efficiency, as demonstrated in Figure 11. We can find that the time overhead is increasing with the height of the RMHT, but the growth rate is very slow. Moreover, the verification operation is one-time and it can be executed off-line. Meanwhile, the time cost is so small that it can be accepted fully, for instance, when the height of the RMHT reaches 80, the time cost is a little more than 1.4 milliseconds. Therefore, it is very efficient for the data owner to verify the data deletion result.

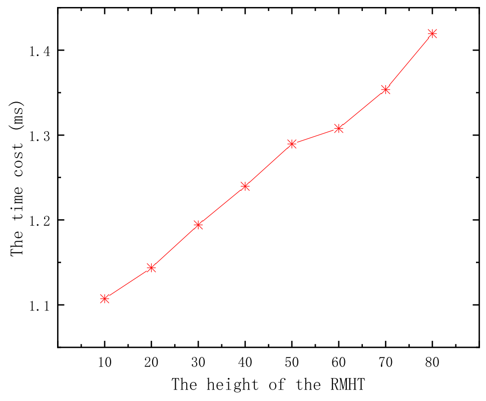

Figure 11: Time cost of deletion result verification

7 Conclusion

In cloud storage, the local data owner doesn't fully believe that the remote cloud server will honestly delete the outsourced data according to the data deletion command. To solve this trust problem, we introduce committed rank-based Merkle hash tree chain (CRMHTC), and then adopt it to design a fine-grained outsourced data deletion scheme. In the proposed scheme, the cloud server stores the data in the rank-based Merkle tree (RMHT) and returns a proof to the data owner to check the storage result. When the data owner will not need the outsourced data anymore, the cloud server deletes the data from the RMHT. By using the CRMHTC, the cloud server generates a corresponding data deletion evidence. After that the data owner can verify the data deletion result to make sure that the data have been indeed deleted. Moreover, different from plenty of the previous solutions, the new proposed scheme doesn't need any trusted third party. Finally, through the security discussion and the simulation results, we show that the proposed scheme can achieve the desired security goals and the efficiency.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 61962015), the Science and Technology Program of Guangxi (No.AB17195045), and the Natural Science Foundation of Guangxi (No.2016GXNSFAA380098). Moreover, the authors gratefully acknowledge the anonymous reviewers for their valuable comments.

References

[1] D. Catalano and D. Fiore, "Vector commitments and their applications," in *International Workshop on Public Key Cryptography*, pp. 55–72. Springer, 2013.

[2] L. Du, Z. Zhang, S. Tan, J. Wang, and X. Tao, "An associated deletion scheme for multi-copy in cloud storage," in *International Conference on Algorithms and Architectures for Parallel Processing*, pp. 511–526. Springer, 2018.

[3] S. L. Garfinkel and A. Shelat, "Remembrance of data passed: A study of disk sanitization practices," *IEEE Security & Privacy*, vol. 1, no. 1, pp. 17–27, 2003.

[4] Neenu Garg and Seema Bawa, "Rits-mht: relative indexed and time stamped merkle hash tree based data auditing protocol for cloud computing," *Journal of Network and Computer Applications*, vol. 84, pp. 1–13, 2017.

[5] B. Hall and M. Govindarasu, "An assured deletion technique for cloud-based iot," in *2018 27th International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–9. IEEE, 2018.

[6] F. Hao, D. Clarke, and A. F. Zorzo, "Deleting secret data with public verifiability," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 6, pp. 617–629, 2015.

[7] C. Liu, R. Ranjan, C. Yang, X. Zhang, L. Wang, and J. Chen, "Mur-dpa: Top-down levelled multi-replica merkle hash tree based secure public auditing for dynamic big data storage on cloud," *IEEE Transactions on Computers*, vol. 64, no. 9, pp. 2609–2622, 2014.

[8] Y. Liu, S. Xiao, H. Wang, and X. Wang, "New provable data transfer from provable data possession and deletion for secure cloud storage," *International Journal of Distributed Sensor Networks*, vol. 15, no. 4, 2019.

[9] Y. Luo, M. Xu, S. Fu, and D. Wang, "Enabling assured deletion in the cloud storage by overwriting," in *Proceedings of the 4th ACM International Workshop on Security in Cloud Computing*, pp. 17–23. ACM, 2016.

[10] Cisco Visual Networking, "Cisco global cloud index: Forecast and methodology, 2016–2021," *White paper. Cisco Public, San Jose*, 2016.

[11] J. Ni, X. Lin, K. Zhang, Y. Yu, and X. S. Shen, "Secure outsourced data transfer with integrity verification in cloud storage," in *2016 IEEE/CIC International Conference on Communications in China (ICCC)*, pp. 1–6. IEEE, 2016.

[12] M. Paul and A. Saxena, "Proof of erasability for ensuring comprehensive data deletion in cloud computing," in *International Conference on Network Security and Applications*, pp. 340–348. Springer, 2010.

[13] J. Reardon, D. Basin, and S. Capkun, "Sok: Secure data deletion," in *2013 IEEE symposium on security and privacy*, pp. 301–315. IEEE, 2013.

[14] V. Turner, J. F. Gantz, D. Reinsel, and S. Minton, "The digital universe of opportunities: Rich data and the increasing value of the internet of things," *IDC Analyze the Future*, vol. 16, 2014.

[15] M. Y. C. Wei, L. M. Grupp, F. E. Spada, and S. Swanson, "Reliably erasing data from flash-based solid state drives," in *FAST*, vol. 11, pp. 105–117, 2011.

[16] R. Xin, M. Ye, J. Wang, K. Hu, and Y. Zhao, "Data deletion method for security improvement of flash memories," *IEICE Electronics Express*, pp. 15–20180152, 2018.

[17] J. Xiong, X. Liu, Z. Yao, J. Ma, Q. Li, K.Geng, and P. S. Chen, "A secure data self-destructing scheme in cloud computing," *IEEE Transactions on Cloud Computing*, vol. 2, no. 4, pp. 448–458, 2014.

[18] L. Xue, J. Ni, Y. Li, and J. Shen, "Provable data transfer from provable data possession and deletion in cloud storage," *Computer Standards & Interfaces*, vol. 54, pp. 46–54, 2017.

[19] L. Xue, Y. Yu, Y. Li, M. H. Au, X. Du, and B. Yang, "Efficient attribute-based encryption with attribute revocation for assured data deletion," *Information Sciences*, vol. 479, pp. 640–650, 2019.

[20] C. Yang, X. Chen, and Y. Xiang, "Blockchain-based publicly verifiable data deletion scheme for cloud storage," *Journal of Network and Computer Applications*, vol. 103, pp. 185–193, 2018.

[21] C. Yang and X. Tao, "New publicly verifiable cloud data deletion scheme with efficient tracking," in *International Conference on Security with Intelligent Computing and Big-data Services*, pp. 359–372. Springer, 2018.

[22] C. Yang, J. Wang, X. Tao, and X. Chen, "Publicly verifiable data transfer and deletion scheme for cloud storage," in *International Conference on Information and Communications Security*, pp. 445–458. Springer, 2018.

[23] C. Yang and J. Ye, "Secure and efficient fine-grained data access control scheme in cloud computing1," *Journal of High Speed Networks*, vol. 21, no. 4, pp. 259–271, 2015.

[24] Changsong Yang, Xiaoling Tao, Feng Zhao, and Yong Wang, "A new outsourced data deletion scheme with public verifiability," in *International Conference on Wireless Algorithms, Systems, and Applications*, pp. 631–638. Springer, 2019.

[25] J. Zhang, B. Wang, D. He, and X. Wang, "Improved secure fuzzy auditing protocol for cloud data storage," *Soft Computing*, vol. 23, no. 10, pp. 3411–3422, 2019.

Biography

Changsong Yang received the B.S. degree in information security from Xidian University, China, in 2014. He was a master student at the School of Telecommunications Engineering of Xidian University, from 2014 to 2015. He is currently a Ph.D candidate in information security at School of Cyber Engineering, Xidian University. His current research interests include cloud computing and security, publicly verifiable data deletion, public key cryptography and Blockchain.

Qiyu Chen is a research scholar in the College of Information and Computer Engineering, Northeast Forestry University.

Yueling Liu is a research scholar in the School of Computer Science and Information Security, Guilin University of Electronic Technology.