SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

December 2018, Vol. 61 122104:1–122104:15
https://doi.org/10.1007/s11432-017-9410-9

Efficient and secure auditing scheme for outsourced big data with dynamicity in cloud

Qingqing GAN, Xiaoming WANG* & Xuefeng FANG

Department of Computer Science, Jinan University, Guangzhou 510632*, China*

Received 7 November 2017/Revised 5 January 2018/Accepted 28 February 2018/Published online 12 November 2018

Abstract Big data offers significant benefits; however, security and privacy problems must be considered, especially with regard to outsourced big data. Auditing outsourced big data is an important factor in security and privacy. However, most of the existing auditing techniques are not suitable for outsourced big data due to their high computational and communication costs at the auditor and the data owner (DO) side. In this paper, we propose an efficient auditing scheme for outsourced big data based on algebraic signatures and an XOR-homomorphic function, that can achieve numerous advantages, such as fewer challenges and proofs, non-block verification, data privacy preservation, and lower computational and communication costs. The proposed scheme enables a trusted third-party auditor, on behalf of DOs, to audit the outsourced data in a cloud. Thus, reducing the computational burden on the DOs. Subsequently, we construct a new data structure called a Record Table (RTable) and extend the basic auditing scheme to support the data dynamic operations. As our extended scheme does not use public key encryption, the entire process of updating the data incurs only a small computational and communication overhead with regard to the auditor, the DOs, and the cloud server. Finally, the proposed basic scheme is proved secure under the security model against various attacks. Analysis of the performance shows that both our basic scheme and extended scheme are highly efficient.

Keywords dynamic auditing, algebraic signature, XOR-homomorphic function, provable security, cloud storage

Citation Gan Q Q, Wang X M, Fang X F. Efficient and secure auditing scheme for outsourced big data with dynamicity in cloud. Sci China Inf Sci, 2018, 61(12): 122104, https://doi.org/10.1007/s11432-017-9410-9

1 Introduction

Big data is attracting attention from all over the world nowadays, and it is forming a close relationship to human life in almost every aspect from information recording, trend analysis, to digital online services [1]. Big data is generally stored in the cloud and managed by cloud service providers. However, such data cannot be guaranteed to be intact and securely stored, since the cloud service provider might be dishonest. For instance, the cloud server (CS) may delete some data that is less frequently accessed, to reduce storage space, conceal data damage or loss events to protect their reputation, and for some other reasons. To prevent such things concerning data owners (DOs), it is very important and urgent to build a mechanism to check the integrity for outsourced big data on a semi-trusted CS. Owning to the characteristics of big data storage in cloud environment, the big data auditing mechanisms should especially consider two factors. (1) Dynamicity: an auditing scheme with this factor enables dynamic

* Corresponding author (email: twxm@jnu.edu.cn)

update for outsourced data in the cloud. (2) Efficiency: an auditing scheme with this factor has minimum computation, storage, and communication overhead. However, most of current data auditing methods are constructed based on public-key cryptography and bilinear pairing (e.g., [2–12]), involving a great number of processes and frequent data transmission. Consequently, these schemes suffer from heavy computational and communication costs to the auditor or the DO, which is not suitable and cannot be directly and effectively used in auditing large-scale data. Moreover, to support dynamic operations on data, most of the existing auditing schemes use various kinds of data structures, such as skip list and B-tree. But these data structures are not capable of achieving large-scale data update operations effectively, since numerous data blocks are required to rebalance frequently when the update happens, which results in heavy overhead on computation at the auditor and CS side [13]. Therefore, it is imperative to design a new auditing technique with dynamicity for large-scale data.

In order to promote efficiency, some data auditing schemes have been formulated [13–17] based on algebraic signature. Algebraic signature was first put forward by Litwin et al. [18], which was in fact one special kind of hash function with algebraic properties and high computational efficiency. To be specific, an algebraic signature owns two primary advantages over other techniques. Firstly, it can perform compression on a large number of file blocks, thus significantly saving the network bandwidth. Secondly, it runs very fast and can be adopted to diagnose data incorrectness, without obtaining the entire file. Schwarz et al. [14] used algebraic signature to check file integrity in large-scale distributed storage systems, leading to the application of algebraic signature more widely. By combining algebraic signature with integrity checking mechanisms, both computational and communication costs can be largely reduced. However, the existing schemes in [14–16] based on algebraic signatures are executed between the DO and the CS. Such a model is unrealistic and impractical for auditing large-scale data, since it needs the DO to challenge and verify data frequently, which also brings a large burden of computational cost imposed on the DO. Under such a situation, auditing by the third party tends to be a natural choice to save the DO's computational cost as well as online burden.

Motivated from the above concerns, we have formulated an efficient and secure auditing scheme for outsourced big data, with dynamicity in the cloud environment. It enables a trusted third-party auditor (TTPA) to check the DO's outsourced data and release the computational burden of the DO. To improve efficiency, our proposed scheme is based on algebraic signature. For the security aspect, we adopt a special kind of pseudo-random function called the exclusive-or (XOR) -homomorphic function to generate a randomized proof. In the current big data era, our scheme can achieve lower computational and communication costs and can be applied to the cloud scenario with flexibility.

1.1 Contributions

In order to verify data integrity more effectively, our paper focuses on the study of auditing for outsourced big data, efficiently and securely in the cloud. The contributions of this paper can be mainly listed as follows.

(1) Based on algebraic signature, we construct an efficient outsourced big data auditing scheme to flexibly realize data integrity checking in cloud environment. Our scheme enables a TTPA, on behalf of DOs, to audit the data outsourced on the CS, thus reducing the computational burden of DOs. With the adoption of algebraic signature, the proposed scheme can achieve numerous advantages, such as a small amount of challenges and responses, non-block verification, and low computational and communication costs. To settle the problem about data privacy, a randomized proof is generated using the XOR-homomorphic function in our scheme, such that the auditor will not be able to learn anything from the data but can audit the data integrity.

(2) The proposed basic scheme can be proved to be secure under the security model against various attacks. Analysis of performance and the results of experiments show that our basic scheme has high efficiency, and can be suitable to audit large-scale data.

(3) By designing a new data structure called Record Table (RTable), we extend our basic scheme to support dynamic operations on outsourced data and allows DOs to achieve data modification, insertion,

appendency, and deletion, without downloading original files from the CS. Without using the public key cryptography technique, our extended scheme incurs only a small computational and communication overhead on the auditor, the DO as well as the CS, when updating data. More importantly, our solution based algebraic signature and XOR-homomorphic function tends to be faster than methods based on cryptography and bilinear pairing, which makes our extended scheme more practical and feasible for large-scale data auditing.

1.2 Related work

Verifying outsourced data integrity in cloud storage remains an active research topic in recent years. In 2007, Ateniese et al. [19] first raised the idea of data auditing and presented two auditing schemes with the use of the homomorphic verifiable tag, which were securely applied to data integrity verification in the cloud environment. However, both schemes caused high computational cost for the adoption of RSA numbering. Meanwhile, Juels and Kaliski [20] put forward a variant type of auditing technique, to check the data integrity depending on forward error-correcting codes and sentinels. Soon afterwards, Shacham and Waters [4] proposed a secure auditing scheme by adopting the BLS homomorphic authentication technique. This new scheme could achieve the aggregation of tags and realize error recovering with the Reed-Solomon code. However, it was not suitable to apply in interactive proof systems since the data may be exposed to attackers in its public verification process. Subsequently, the thought of data auditing was adopted widely and many constructions were built. For example, Wang et al. [5] designed a remote data checking protocol that relies on bilinear aggregate signature and Merkle hash tree. They extended their scheme in [3] and found a solution to enable privacy-preserving public auditing as well as batch verification in multi-user environments. However, both schemes brought high computational overhead and were impractical for large-scale files in cloud. To settle some security drawbacks, Yu et al. [21] put forward an improved remote data auditing scheme, which was based on the random sampling technique in [19], to ensure the integrity of file blocks. But its TagBlock and ProofVerify phases have incurred heavy computational costs.

As for the dynamic auditing methods, Ateniese et al. [22] formulated a dynamic auditing scheme to achieve the add, modify, and delete operations on data. However, when the updating happened, all tokens needed to be re-computed by the DO leading to a large computational cost; thus this method was inefficient for big data storage. Shortly after, Erway et al. [23] constructed an efficient auditing scheme supporting dynamic data update. However, the integrity of each individual file block cannot be checked in this scheme. Later, Cash et al. [24] built a dynamic auditing method relying on the ORAM technique, but it also had the shortcoming of great computational cost for the DO and the server. Yang et al. [25] designed an auditing scheme with privacy-preservation for cloud storage systems and extended their protocol, enabling the data dynamic operations. Shen et al. [26] constructed a public auditing framework for large-scale data, where data dynamics were supported. In their design, the dynamic structure was made up of a location array and a doubly linked table. However, both schemes in [25,26] included bilinear pairing operations, which brought about high computational overhead.

To improve the efficiency in the integrity verification, constructing data auditing schemes based on algebraic signature have been proposed [13–17]. Chen [15] employed algebraic signatures to create a data auditing scheme. However, this protocol cannot resist the replay attack and is vulnerable to deletion attack from a malicious server [21]. In addition, the third-party auditor was not introduced in this scheme, bringing great computational burden to the DO. Before long, a new data checking protocol based on algebraic signature was proposed for the cloud environment in [17]. The proposed scheme allowed a third-party auditor to audit data integrity and supported an unlimited number of verifications. Although it was efficient with a smaller overhead, the data could be revealed since it cannot be protected from attacks [27]. Recently, Sookhak et al. [13] introduced another kind of data auditing technique, which was based on algebraic signature and can perform remote data checking efficiently. However, their scheme is insecure and susceptible to the replay attack. Thus, the CS can generate a valid proof without accessing the actual data from the DO. Besides, in their challenge phase, the challenge messages do not

contain an extra random value. Therefore, the proof calculated by the CS is not randomized and the CS can always send the same proof to DO. Furthermore, it only checks the correctness for the verification process, but does not provide a specific security proof under any security model.

So far, most of the auditing schemes either ignore the data privacy issue or involve a high computational cost, which is not secure or efficient enough for large-scale files in cloud environment. Therefore, according to the characteristics of cloud storage, the research and the establishment of an efficient and secure data auditing scheme for outsourced big data are very necessary and urgent, which can have theoretical significance as well as application value.

1.3 Organization

The rest of paper is organized as follows: Section 2 introduces some basic knowledge, including the definition and security model for data auditing, algebraic signature, and XOR-homomorphic function. Section 3 details the system model and demonstrates our construction of efficient outsourced big data auditing scheme. Section 4 shows the evaluation of our efficient data auditing scheme, containing security analysis and performance analysis. Thereafter, we extend our basic scheme to support dynamic operations, including data modification, insertion, appendency, and deletion, as well as discuss performance in Section 5. Finally, the conclusion of this paper is drawn in Section 6.

2 Preliminaries and notations

This section presents the main concepts underlying the proposed scheme.

2.1 Definition of data auditing scheme

An outsourced big data auditing scheme consists of five polynomial-time algorithms: Setup, TagBlock, Challenge, Proof, and Verification, which are the same as [21].

(1) Setup(1^λ): The Setup algorithm takes a security parameter λ as input. It will randomly generate public parameters params, and the secret key sk.

(2) TagBlock(params, sk, F): The TagBlock algorithm inputs params, sk, as well as a file F. We assume that the file F is split into n blocks, each individual file block denoted as $F[i]$ for $1 \leqslant i \leqslant n$. It outputs a tag T_i for each block.

(3) Challenge: The Challenge algorithm outputs a challenge message chal for the blocks to be checked whether they are stored in the CS.

(4) Proof(params, F, T_i, chal): The Proof algorithm inputs params, blocks of file F, a challenge chal, and tags corresponding to the blocks in F. It outputs the proof message prf.

(5) Verification(params, sk, chal, prf): The Verification algorithm inputs params, sk, a challenge chal, as well as the proof message prf. It outputs 0 or 1 to indicate data integrity.

2.2 Security model of the data auditing scheme

We now present the security model for the outsourced big data auditing scheme. The game is carried out between a polynomial-time adversary and a challenger, detailed as the following [19, 28].

(1) Setup: The challenger runs Setup(1^λ) and obtains public parameter params and the secret key sk. Then the challenger keeps sk secret and delivers params to the adversary.

(2) Query: The adversary requests tag queries adaptively. For each query, the adversary chooses a block $F[i]$ and transmits the block to the challenger. By running TagBlock(params, sk, F), the challenger computes the corresponding tag T_i and returns T_i to the adversary.

(3) Challenge: The challenger generates a challenge chal.

(4) Forge: The adversary produces a proof message prf and sends prf to the challenger.

If the Verification(params, sk, chal, prf) algorithm outputs 1, it means that the adversary wins the game. We define that a data auditing scheme is secure under attacks, if the probability of winning the aforementioned game for all polynomial-time adversaries is negligible.

2.3 Algebraic signature

Algebraic signature can be used to calculate the signature of unseen messages in a limited way. Its properties can be explained as follows [13]. Suppose there are two files: a file F with length r and a file G. The file F is split into n data blocks $(F[1], F[2], \ldots, F[n])$.

Proposition 1. The algebraic signature of the file F is computed as

$$S_\gamma(F) = \sum_{i=1}^{n} F[i] \cdot \gamma^{i-1},$$

where γ is a primitive element in the Galois field and is comprised of a vector of n non-zero elements as $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_n)$.

Proposition 2. The algebraic signature of the concatenation between the file F and the file G is computed as

$$S_\gamma(F \parallel G) = S_\gamma(F) \oplus r^\gamma S_\gamma(G).$$

Proposition 3. The algebraic signature of a combination of the file F and the file G equals to the combination of algebraic signatures of each file, denoted as

$$S_\gamma(F \oplus G) = S_\gamma(F) \oplus S_\gamma(G).$$

Proposition 4. The algebraic signature of a combination of n blocks in file F is equal to the combination of algebraic signatures of each of the corresponding blocks, denoted as

$$\sum_{i=1}^{n} S_\gamma(F[i]) = S_\gamma\left(\sum_{i=1}^{n} F[i]\right).$$

The main reason for our adoption of algebraic signature is that, the tag in our scheme can be designed as the algebraic signature of a file block. Featured with its efficient and algebraic character, algebraic signature can achieve the compression of a large block into a short bit string. Although it is only probabilistically secure, it can be regarded as an ideal method for verifying large amounts of outsourced data, with minimal network overhead [15]. Furthermore, the bit string can be set large enough to make an accidental collision attack extremely unlikely. Therefore, algebraic signature's advantages of low network load, reasonable computational cost, and resistance to malicious modification makes it quite suitable for integrity verification of remote cloud data.

2.4 XOR-homomorphic function

The XOR-homomorphic function refers to the pseudo-random function with the property of XOR-homomorphism, which can enhance the protection against data leakage and ensure data privacy. Its properties can be shown as below [29].

Proposition 5. For an XOR-homomorphic function f, the XOR operation by two inputs through f is equal to the XOR operation by function f with each input, denoted as

$$f(x_1 \oplus x_2) = f(x_1) \oplus f(x_2).$$

Proposition 6. The permutation on bits of data is one instance of XOR-homomorphic function. With the permutation keys (k_1, k_2), we have the following equation:

$$f_{k_1}(x_1) \oplus f_{k_2}(x_2) = f_{k_1 \oplus k_2}(x_1 \oplus x_2).$$

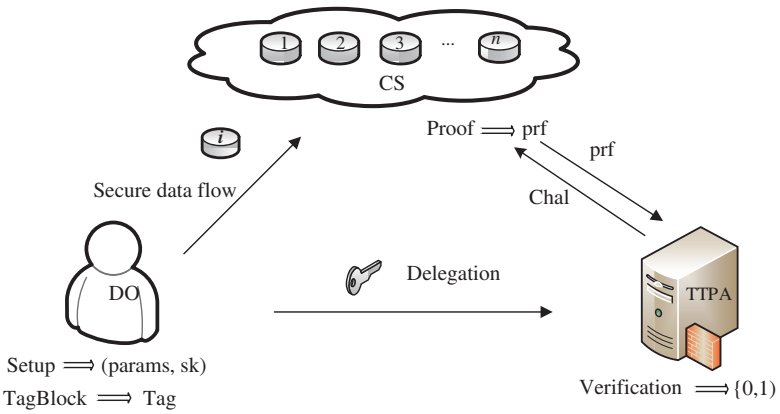

Figure 1 (Color online) The architecture of outsourced big data auditing in cloud storage.

Using the above XOR-homomorphic function, the random permutation keys and two inputs are XORed together and a randomized output is produced. Such an operation can prevent any eavesdroppers from obtaining any bit information about the input data without permutation keys, even by analyzing the traffic. As the bit permutation possesses the property of XOR-homomorphism, a secure permutation algorithm can be used to guarantee the security of the schemes. For example, the famous Fisher-Yates shuffle [30] can generate distributed permutations uniformly with efficiency at each session, thus enabling session security. Equipped with the above properties, the XOR-homomorphism function allows secure verification and protects the information randomness from attackers.

3 Main construction

In this section, a formal description is elaborated for the outsourced big data auditing scheme.

3.1 System model

For cloud environment, the architecture of efficient outsourced big data auditing in cloud storage contains three entities, which is demonstrated in Figure 1. The DO is an individual user or a company who has outsourced the data to the CS, thus mitigating the local storage cost; the CS has substantial storage capacity as well as computational power and is semi-trusted; the TTPA can be delegated by the DO to process data possession verification, on behalf of the DO upon request, thereby reducing the computational burden of the DO.

When the DO wants to store multiple files on CS, he will first run the Setup algorithm to get public system parameter params and the secret key sk, and keep sk secret. Then he runs the TagBlock algorithm to generate a tag for every file block, and uploads these files together with tags to the CS in a secure data flow. Once the DO hopes to check whether these files are stored on the CS in an intact manner, he can delegate the checking operation to the TTPA, with a secret key passed to the TTPA through a secure channel. The TTPA invokes the Challenge algorithm to generate chal and delivers it to the CS. By receiving challenge message chal, the CS will invoke the Proof algorithm and return the proof information prf to the TTPA. Finally, the TTPA will call the Verification algorithm to check the integrity of these files. If the Verification algorithm outputs 1, it means prf is valid and the integrity of the stored file is preserved.

3.2 Outsourced big data auditing scheme

In preliminary processing, we adopt the data fragment technique to split a file F as n blocks and split every block as s sectors. For the sake of security, the size of blocks and sectors is limited by the security

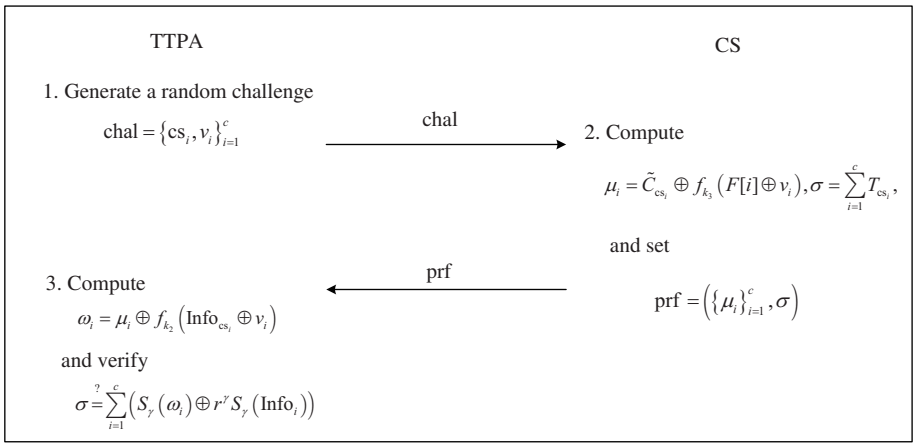

Figure 2 The process of the outsourced big data auditing scheme.

parameter. Considering simplicity, here we only refer to each individual file block denoted as $F[i]$, while $1 \leqslant i \leqslant n$. The outsourced big data auditing scheme consists of five algorithms, listed as follows.

(1) Setup(1^λ): The prime q is determined by the security parameter λ. The DO generates secret keys $\{k_1, k_2, k_3\}$ randomly from Z_q^*, and defines a XOR-homomorphic function $f : Z_q^* \times Z_q^* \to Z_q^*$. The DO randomly chooses a primitive element γ in the Galois field and defines an algebraic signature S_γ. The public parameter params are $\{q, f, S_\gamma\}$, while the secret key is sk $= \{k_1, k_2, k_3\}$.

(2) TagBlock(params, sk, F): The DO chooses n random values $\{R_i\}_{i=1}^n$ from Z_q^*, and then represents each block with an information $\text{Info}_i = \{\text{Ind}_i \parallel V_i \parallel \text{TS}_i\}$, where Ind_i denotes the index, V_i stands for the version with an initialized value ($V_i = 1$), and TS_i means the time stamp. The DO computes

$$C_i = f_{k_1 \oplus k_2 \oplus k_3}(F[i] \oplus R_i), \quad \widetilde{C}_i = f_{k_1}(\text{Info}_i \oplus R_i),$$

and the corresponding tag for each file block is $T_i = S_\gamma(C_i \parallel \text{Info}_i)$. The DO sends $\{F[i], T_i, \widetilde{C}_i\}_{i=1}^n$ to the CS and then deletes the local copy of the file F. At same time, the DO sends k_2 and $\{\text{Info}_i\}_{i=1}^n$ to the TTPA and k_3 to the CS through a secure channel.

(3) Challenge: The TTPA selects c random values $\{v_i\}_{i=1}^c$ with $c \leqslant n$, and sends a challenge request chal $= \{\text{cs}_i, v_i\}_{i=1}^c$ to the CS, where cs_i denotes the challenged block.

(4) Proof(params, F, T_i, chal): When the CS receives the challenge message chal, it will compute

$$\mu_i = \widetilde{C}_{\text{cs}_i} \oplus f_{k_3}(F[i] \oplus v_i), \quad \sigma = \sum_{i=1}^{c} T_{\text{cs}_i},$$

and send prf $= (\{\mu_i\}_{i=1}^c, \sigma)$ to the TTPA as the proof message.

(5) Verification(params, sk, chal, prf): Upon receiving the proof message prf, the TTPA first computes $\widehat{C}_{\text{cs}_i} = f_{k_2}(\text{Info}_{\text{cs}_i} \oplus v_i)$. Then, the TTPA calculates $\omega_i = \mu_i \oplus \widehat{C}_{\text{cs}_i}$. Finally, the TTPA verifies the integrity of the file F as follows:

$$\sigma \stackrel{?}{=} \sum_{i=1}^{c}(S_\gamma(\omega_i) \oplus r^\gamma S_\gamma(\text{Info}_i)).$$

If the above equation holds, it demonstrates that the file is integrated and well stored on CS; otherwise, the data has been damaged. The process for Challenge, Proof and Verification in the outsourced big data auditing scheme is shown in Figure 2.

The correctness of the scheme is illustrated as follows:

$$\begin{aligned}&\sum_{i=1}^{c}(S_\gamma(\omega_i) \oplus r^\gamma S_\gamma(\text{Info}_i))\\ &\quad = \sum_{i=1}^{c}(S_\gamma(\omega_i \parallel \text{Info}_i)) = \sum_{i=1}^{c}(S_\gamma((\mu_i \oplus \widehat{C}_{\text{cs}_i}) \parallel \text{Info}_i))\end{aligned}$$

$$= \sum_{i=1}^{c} (S_\gamma((f_{k_1}(\mathrm{Info}_{\mathrm{cs}_i} \oplus R_{\mathrm{cs}_i}) \oplus f_{k_3}(F[\mathrm{cs}_i] \oplus v_i) \oplus f_{k_2}(\mathrm{Info}_{\mathrm{cs}_i} \oplus v_i)) \parallel \mathrm{Info}_i))$$

$$= \sum_{i=1}^{c} (S_\gamma(f_{k_1 \oplus k_2 \oplus k_3}(F[\mathrm{cs}_i] \oplus R_{\mathrm{cs}_i}) \parallel \mathrm{Info}_i))$$

$$= \sum_{i=1}^{c} (S_\gamma(C_{\mathrm{cs}_i} \parallel \mathrm{Info}_i)) = \sum_{i=1}^{c} T_{\mathrm{cs}_i} = \sigma.$$

4 Evaluation

To evaluate our proposed outsourced big data auditing scheme, we analyze it from two aspects: security analysis and performance analysis.

4.1 Security analysis

With the following theorem, our scheme is proved to be secure by the security model from [19] and games in [21]. If the scheme is secure, it means that a polynomial-time adversary, without possessing the whole file, cannot forge a valid proof to pass the integrity verification.

Theorem 1. If the XOR-homomorphic function is secure, then it means that no polynomial-time adversary can break our outsourced big data auditing scheme with a non-negligible probability. That is to say, a corrupted proof will not get through in the verification process, unless the proof is generated honestly by the Proof algorithm in our scheme.

Proof. Initially, we imitate a series of games and define each game with limited indistinguishable changes in the adversary's behavior.

Game 0. As the initial game, Game 0 remains as the original security model described in Subsection 2.2.

Game 1. In this game, the challenger adopts random values in Z_q^* to replace the outputs of the XOR-homomorphic function f, noticing that $C_i = f_{k_1 \oplus k_2 \oplus k_3}(F[i] \oplus R_i)$ and $\widetilde{C}_i = f_{k_1}(\mathrm{Info}_i \oplus R_i)$ are originally used for verification of the file blocks. The challenger generates two random values $a, b \in_R Z_q^*$ to replace the corresponding $C_i, \widetilde{C}_i$ computed by the above equations. Then, the challenger inserts an entry $(F[i], \mathrm{Info}_i, k_1, k_2, k_3, a, b)$ into the table T-List to answer the adversary's tag queries, where T-List is initially empty. If the adversary can distinguish a given challenged tag from the random value, then the challenger will report failure and abort. Once the adversary has finished the TagBlock queries phase, the challenger will store the table T-List about the queries and the corresponding answers, to keep consistency.

If the probability of success for the adversary between Games 0 and 1 is non-negligible, we can use the adversary's ability to break the semantic security of the XOR-homomorphic function f. To connect the gap between Games 0 and 1, we explain that it will lead the reduction to the security loss with $\frac{1}{\mathrm{mq}_t}$, in which m stands for the number of blocks that the adversary requests to have maintained, and q_t denotes the number of tag queries from the adversary. Hence, the adversary never distinguishes a valid tag generated by the TagBlock algorithm from the random value in the table T-List, except with the negligible probability $\frac{1}{\mathrm{mq}_t}$, which means that the probability for the challenger aborting can be negligible.

Game 2. In this game, the challenger executes our scheme initiated by the adversary, which is different from that in Game 1. In such an outsourced big data auditing scheme, the challenger generates a challenge chal just like before. But the challenger makes the verification about the response from the adversary in a way different from the original verification process.

To be specific, in the Forge phase, if values sent from the adversary are the same as the query responses, the challenger will accept the adversary's answers. Once in the whole process, the proof received from the adversary enables to get through the Verification algorithm successfully, and is not generated from an honest server; then the challenger reports failure.

The distinction between Games 1 and 2 appears only when in some processes, the response of the adversary satisfies the Verification algorithm, while it has not been produced from the challenger acting as an honest server. We analyze that this will happen at a negligible probability.

To show the difference in probabilities of Games 1 and 2, the following conception is declared. Suppose that the challenger sends chal $= \{\mathrm{cs}_i, v_i\}_{i=1}^{c}$ to the adversary that causes aborts, the adversary's answer to the query is $(\{\mu_i^*\}_{i=1}^{c}, \sigma^*)$. The adversary's response can pass integrity verification, only if the following equation holds:

$$\sigma^* = \sum_{i=1}^{c} (S_\gamma(\mu_i^* \oplus \widehat{C}_{\mathrm{cs}_i}) \oplus r^\gamma S_\gamma(\mathrm{Info}_i)).$$

Notice that for $1 \leqslant i \leqslant c$, $\widehat{C}_{\mathrm{cs}_i} = f_{k_2}(\mathrm{Info}_{\mathrm{cs}_i} \oplus v_i)$. So we get $\sigma_i^* = (S_\gamma(\mu_i^* \oplus \widehat{C}_{\mathrm{cs}_i}) \oplus r^\gamma S_\gamma(\mathrm{Info}_i))$. If the response is obtained from an honest server, that is $(\{\mu_i\}_{i=1}^{c}, \sigma)$, where $\mu_i = \widetilde{C}_{\mathrm{cs}_i} \oplus f_{k_3}(F[i] \oplus v_i)$, $\sigma = \sum_{i=1}^{c} T_{\mathrm{cs}_i}$, the following verification equation will be satisfied:

$$\sigma = \sum_{i=1}^{c} (S_\gamma(\mu_i \oplus \widehat{C}_{\mathrm{cs}_i}) \oplus r^\gamma S_\gamma(\mathrm{Info}_i)).$$

From the above equation, we have $\sigma_i = S_\gamma(\mu_i \oplus \widehat{C}_{\mathrm{cs}_i}) \oplus r^\gamma S_\gamma(\mathrm{Info}_i)$. If $\mu_i^* = \mu_i$, then $\sigma_i^* = \sigma_i$, $\sigma^* = \sigma$, which is contradictory to our assumption above. Therefore, we define $\Delta\sigma_i = \sigma_i^* \oplus \sigma_i$ and $\Delta\mu_i = \mu_i^* \oplus \mu_i$. Let we make the XOR operation on the verification equation for σ_i^* and σ_i. The result is shown as

$$\Delta\sigma_i = S_\gamma(\Delta\mu_i).$$

The bad case happens only when some $\Delta\mu_i$ are not zero, which shows that the answer sent from the adversary is not the same as the response from an honest server. Now we take the values $\Delta\mu_i$ and $\Delta\sigma_i$ into consideration. The above equation holds for some specific values in an interaction with the probability is $\frac{1}{q}$. Then we obtain that, the maximum probability, satisfying the equation for a non-zero number of values, is $\frac{q_s}{q}$, in which q_s stands for the number of data auditing interactions in the game. Therefore, the adversary cannot compute a valid proof different from an honest server's response, except with a negligible probability $\frac{q_s}{q}$, meaning that the probability for the challenger aborting is negligible.

Hence, assuming XOR-homomorphic function is secure, compared to Game 0, only a negligible difference in probability exists for a polynomial-time adversary for success in Game 2. Therefore, we have completed the proof of Theorem 1.

4.2 Performance analysis

Now, we evaluate the performance for our outsourced big data auditing scheme, which contains analyzing the probability of misbehavior detection, making comparison with several related schemes, and showing experimental results.

Our scheme is conducted by a random sampling technique depicted in [19], which can greatly lessen the workload on the CS. Suppose the input file F is to be divided as n blocks, the sampling strategy will randomly select c blocks as a challenge to perform the protocol process. Accordingly, the probability of misbehavior detection in our scheme can be evaluated as

$$p_x = P\{x \geqslant 1\} = 1 - P\{x < 0\} = 1 - \frac{\prod_{i=0}^{i=c-1}(n-x-i)}{\prod_{i=0}^{i=c-1}(n-i)},$$

where x denotes the number of the challenged blocks from the TTPA that matches the corrupted blocks from cloud server, so that p_x stands for the possibility that at least one challenged block by the TTPA can match one corrupted block. Then we have

$$1 - \left(1 - \frac{x}{n}\right)^c \leqslant p_x \leqslant \left(1 - \frac{x}{(n-c+1)}\right)^c.$$

Table 1 Comparison with related schemes

Scheme	Cryptography based on	Server computation complexity	Verifier computation complexity	Communication complexity	Verifier storage complexity	Dynamic auditing	Third-party auditor
PDP [19]	Public-key	$O(1)$	$O(1)$	$O(1)$	$O(1)$	No	No
DPDP [23]	Public-key	$O(\log n)$	$O(\log n)$	$O(\log n)$	$O(1)$	Yes	No
PADD [5]	Public-key	$O(\log n)$	$O(\log n)$	$O(\log n)$	$O(1)$	Yes	Yes
RDPC [21]	Symmetric-key	$O(\log n)$	$O(\log n)$	$O(\log n)$	$O(1)$	No	No
Our basic scheme	Symmetric-key	$O(1)$	$O(1)$	$O(1)$	$O(1)$	No	Yes

Thus, once the CS corrupts a small part of the file, the number of challenged blocks need to grow greatly, for reaching a high probability in error detection. As evaluated in [19], if the corrupted proportion of the whole file is 1%, we need 300 challenged blocks to get the probability satisfying $p_x \geqslant 95\%$, and 460 blocks for satisfying $p_x \geqslant 99\%$. Hence, even when the number of corrupted blocks is small, our outsourced big data auditing scheme can achieve a high probability for detecting errors.

Regarding the cost for computation, storage, and communication, we compare our basic scheme with the schemes in [5, 19, 21, 23]. To be specific, the comparison is based on whether public-key cryptography or symmetric-key cryptography is used, the computation complexity at the server side and verifier side, the communication complexity, the storage complexity at the verifier side, whether dynamic auditing is supported, and whether third-party auditor is introduced, as shown in Table 1. Note that n represent the maximum number of encrypted files in the system; server computation cost denotes the cost of generating proof and the verifier computation cost denotes the cost of the verification process.

As can be seen from Table 1, our outsourced big data auditing scheme and the schemes in [5, 19, 21, 23] all keep the storage complexity at the verifier side at constant size. However, the computation complexity for server, verifier and communication complexity are $O(1)$ in our scheme, which is the same as scheme in [19], and better than schemes in [5, 21, 23] with a complexity of $O(\log n)$. On the other hand, schemes in [5, 19, 23] are based on public-key cryptography, whereas scheme in [21] and our scheme are based on symmetric-key cryptography. It is known that symmetric-key cryptography tends to have lower computational overhead compared to that of public-key cryptography. Considering dynamic auditing, we can see schemes in [5, 23] can support data update operations, whereas schemes in [19, 21] and our basic scheme cannot. However, we will extend our basic scheme to support dynamic auditing, which is discussed in Section 5. Moreover, we can see from Table 1 that schemes in [19, 21, 23] have not introduced a third-party auditor, whereas scheme in [5] and our basic scheme can support the third-party auditor to perform data auditing. With the help of the third-party auditor, the DO's computational overhead can be greatly reduced. Above all, our scheme is superior to schemes in [5, 19, 21, 23], with less costs for computation, storage and communication.

To show the efficiency of our scheme, we conduct an experiment for integrity verification on data blocks. The experiment is implemented with an Intel Core i7-6700M CPU @3.40 GHz and 4.00 GB RAM. We use the development tool Eclipse and complete the programming in JAVA. Algebraic signature in our scheme is achieved by the Galois field GF(2^{16}) calculations. As for the XOR-homomorphic function, we adopt the Fisher-Yates Shuffle to realize secure permutations. We divide all files into 16 blocks and 16 sectors. The sectors' size varies from 1024 to 8192 bits. Then, we record the average time for the system Setup algorithm, tag generation in the TagBlock algorithm, proof generation in the Proof algorithm, and verification phase in the Verification algorithm. The experimental results are illustrated in Table 2.

From Table 2, we can see that the average time of the Setup algorithm appears to be independent of the increasing size of sectors, for it only generates public parameters and the secret key. The average time for the TagBlock algorithm in the simulation varies almost linearly with the increasing size of sectors, as the computational cost in the TagBlock algorithm is related to the size of sectors in our scheme. By contrast, the average time for the Proof and the Verification algorithm increased slightly with the change in size of sectors. To make the results more intuitive, we reflect the TagBlock, Proof and Verification algorithm in Figure 3. The experiment shows that our scheme tends to be highly efficient and practical.

Table 2 Performance of our basic scheme for 16 blocks × 16 sectors

Size of sectors (bits)	Time for Setup (ms)	Time for TagBlock (ms)	Time for Proof (ms)	Time for Verification (ms)
1024	≈ 0.32	2.81	1.65	1.59
2048		3.03	1.68	1.63
4096		3.19	1.69	1.65
8192		3.46	1.72	1.69

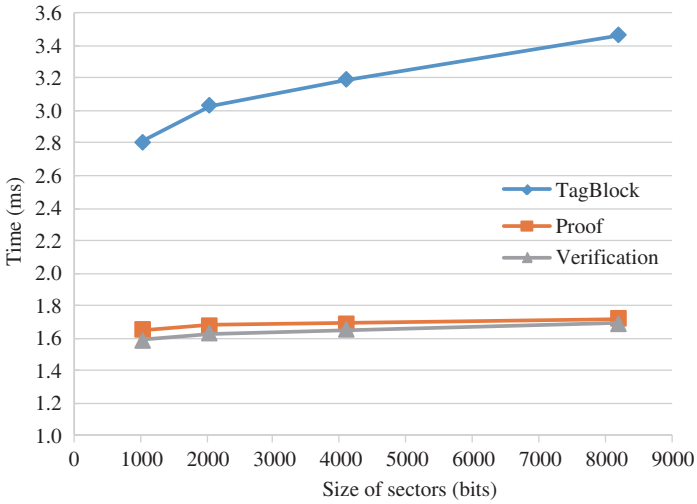

Figure 3 (Color online) Computation cost for 16 blocks × 16 sectors.

5 Dynamic data auditing

When the DO must update the outsourced data, it is vital for outsourced big data auditing schemes to support dynamic data auditing. Inspired by scheme in [25], we construct our data structure RTable to realize the frequent data update requirements from DOs. It can allow the DO to achieve modification, insertion, appendency, and deletion for large-scale data, without downloading original files from the cloud server. Without using the public key cryptography technique, our scheme can perform dynamic update operations effectively.

RTable contains two columns: Record and Block information Info_i, which is stored at the TTPA. It can not only realize dynamic auditing but also prevent the replay attack and forge attack during dynamic operations. When the update operation happens, the TTPA will manage the RTable for the data dynamic operations. Table 3(a) shows the initial RTable. Since data block appendency means that the DO can append a new data block to the end of the file, we consider it as a special type of data insertion operation. Therefore, we incorporate and discuss the data block appendency in the part of data block insertion.

5.1 Data block modification

When the file block $F[i]$ is modified as $F'[i]$ while $1 \leqslant i \leqslant n$, the DO invokes the Modify algorithm to compute a new tag T_i'. Then the DO sends T_i' to the Cloud Server (CS). Meanwhile, the DO sends block information to the TTPA. The algorithm Modify(params, sk, i, $F'[i]$) is shown as follows.

(1) The DO will increase the V_i to V_i' where $V_i' = (V_i + 1)$. Then, the DO updates $\text{Info}_i' = \{\text{Ind}_i \parallel V_i' \parallel \text{TS}_i'\}$, where TS_i' is a new time stamp. The DO chooses a random value R_i' from Z_q^* for the update block. The DO computes

$$C_i' = f_{k_1 \oplus k_2 \oplus k_3}(F'[i] \oplus R_i'), \quad \widetilde{C'}_i = f_{k_1}(\text{Info}_i' \oplus R_i'),$$

and the corresponding tag of each block is $T_i' = S_\gamma(C_i' \parallel \text{Info}_i')$. The DO sends $\{F'[i], T_i', \widetilde{C}_i'\}$ to CS and sends Info_i' to the TTPA, and then deletes its local copy of the file F.

Table 3 Record table (RTable) for block information

(a) Initial Rtable

Record	Block information
1	Info_1
2	Info_2
3	Info_3
$\vdots$	$\vdots$
n	Info_n

(b) Update for modifying $F[2]$ to $F'[2]$

Record	Block information
1	Info_1
2	Info'_2
3	Info_3
$\vdots$	$\vdots$
n	Info_n

(c) Update for inserting F' before $F[2]$

Record	Block information
1	Info_1
2	Info_{n+1}
3	Info_2
$\vdots$	$\vdots$
$n+1$	Info_n

(d) Update for deleting $F[2]$

Record	Block information
1	Info_1
2	Info_3
3	Info_4
$\vdots$	$\vdots$
$n-1$	Info_n

(2) After receiving $\{F'[i], T'_i, \widetilde{C}'_i\}$, the CS will replace the original file block and tag, with the new one.

(3) When the TTPA receives the new block information Info'_i, he will modify the original block information as Info'_i for the corresponding record at RTable. Take $i = 2$ as an example, when the DO wants to update $F[2]$, the modification process can be described as Table 3(b).

5.2 Data block insertion

In the data insertion phase, suppose the DO wants to insert a new file block F' before the file block $F[i]$, he will invoke the Insert algorithm to obtain a new tag for the file block F'. The algorithm Insert(params, sk, i, F') is shown as follows.

(1) The DO defines $\text{Info}_{n+1} = \{\text{Ind}_{n+1} \parallel V_{n+1} \parallel \text{TS}_{n+1}\}$, meaning the index $n+1$, an initialized version value $V_{n+1} = 1$, and the current time stamp TS_{n+1}. The DO chooses a random values R_{n+1} from Z_q^* and computes

$$C_{n+1} = f_{k_1 \oplus k_2 \oplus k_3}(F' \oplus R_{n+1}), \quad \widetilde{C}_{n+1} = f_{k_1}(\text{Info}_{n+1} \oplus R_{n+1}),$$

and the corresponding tag $T_{n+1} = S_\gamma(C_{n+1} \parallel \text{Info}_{n+1})$. The DO sends $\{F', T_{n+1}, \widetilde{C}_{n+1}\}$ to the CS, and sends Info_{n+1} to the TTPA, and then deletes its local copy.

(2) After receiving $\{F', T_{n+1}, \widetilde{C}_{n+1}\}$, the CS inserts and stores the new file block and tag.

(3) When the TTPA receives the new block information Info_{n+1}, he will insert the Info_{n+1} at the record value i, and the original records valued j ($j \geqslant i$) at RTable will be moved backward, thus increasing their values by 1. Take $i = 2$ as example, when the DO tries to add a new file block before $F[2]$; the insertion process can be described as Table 3(c).

As for the data block appendency, data blocks can be allowed to be inserted into the end of the file, which can be regarded as a special kind of data block insertion operation. To be exact, the data block appendency operation has the same steps (1) and (2) of the Insert algorithm, while it has only slightly differences in step (3) of the Insert algorithm. Therefore, when the data block appendency happens, the DO first invokes step (1) of the Insert algorithm and then CS runs step (2) of the Insert algorithm. For the step (3), when the TTPA receives the new block information Info_{n+1}, he will insert a new row $(n+1, \text{Info}_{n+1})$ to the end of RTable, and keep the other original records unchanged. Therefore, data block appendency has been achieved.

Table 4 Comparison with related schemes [a)]

Scheme	Dynamic auditing	Against replay attack	Data structure	Data structure management	Dynamic operation computation
MB-CMCPDP [31]	Yes	Yes	MT	DO/Trusted verifier	$(H + s\text{Mul} + (s+1)\text{Exp})n$
CBSS [32]	Yes	Yes	BST	TTP	$(H + \text{Enc}_{\text{SE}} + \text{FR} + \text{Dec}_{\text{BE}})n$
DAP [25]	Yes	Yes	ITable	Auditor	$(H + (s+1)\text{Mul} + (s+1)\text{Exp})n$
RDA [13]	Yes	No	DCT	DO/TPA	$(2\text{AS}^{\ddagger})n$
Our extended scheme	Yes	Yes	RTable	TTPA	$(\text{AS}+2\text{XOR-HF}^{\ddagger})n$

a) H: Hash function; Mul: multiplication; Exp: exponentiation; Enc_{SE}: encryption algorithm by symmetric encryption; FR: forward rotation; Dec_{BE}: decryption algorithm by broadcast encryption; AS: algebraic signature; XOR-HF: XOR-homomorphic function; ‡: cost of $\oplus$ operation which is usually regarded as negligible.

5.3 Data block deletion

When data block deletion occurs, the DO sends the request to the CS to delete data and tag. In the meantime, the TTPA updates RTable correspondingly. The algorithm Delete(params, i) is shown as follows:

(1) The DO applies to delete the file block $F[i]$ with index i.

(2) The CS deletes the file block $F[i]$ and tag T_i.

(3) The TTPA will update RTable by delete Info_i, and then the original record after the ith position in RTable will be moved forward in order, thus decreasing their values by 1. Take $i = 2$ as an example, when the DO wants to remove $F[2]$, the deletion process can be described as Table 3(d).

5.4 Performance discussion

In the above, we showed our construction for dynamic data auditing, which can achieve remote data modification, insertion, appendency, and deletion respectively. To demonstrate the merits of the extended scheme, we compare with schemes in [13, 25, 31, 32] and the results are displayed at Table 4.

From Table 4, we can see all the compared schemes enable dynamic data auditing. However, different data structures are designed in these schemes to realize a dynamic function. For example, scheme in [31] adopted a map-version table (MT) to support remote data update, which consists of three columns: serial number (SN), block number (BN), and key version (KV). The values in MT will be changed due to different dynamic operations. To improve scheme in [31], a dynamic data structure named block status table (BST) was designed in [32]. BST is almost the same as MT, except the newly proposed one had an extra control parameter: ctr, which is used to mark the revocation process. Scheme [25] introduced an index table called ITable to prevent the replay attack and forge attack during dynamic operations. In their method, ITable contains four components: Index, block number (B_i), version number (V_i), and time stamp (T_i). When data update happens, the TPA will update the ITable to achieve data dynamicity with efficiency and security. Scheme [13] adopted a data structure called Divide and Conquer Table (DCT) to achieve data update function efficiently. DCT contains two components: logical index (L_i), and version number (V_i). However, this scheme is insecure and susceptible to the replay attack. As for our scheme, we construct a data structure named RTable to realize the frequent data update requirements from DOs. In our scheme, we set RTable containing only two columns: Record and Block information (Info_i), thus not only saving the storage cost but also preventing the replay attack and the forge attack. To be specific, we concatenate block index (Ind_i), version (V_i), and time stamp (ST_i) together as Info_i. For the data structure management, the data structures in schemes in [25, 32] and our extended scheme are managed by the third party, while schemes in [13, 31] set the DO or other verifiers as the data structure manager. As for the computational cost for dynamic data operations, we mainly measure about the tag generation process in the modification, insertion and appendency algorithm. Let n represent the number of update blocks, and each block is divided into s sectors. The computational cost for dynamic operations in schemes in [13, 25, 31, 32] and our extended scheme are shown at Table 4. Above all, our scheme is superior to schemes in [13, 25, 31, 32], both secure and efficient on dynamic auditing.

6 Conclusion

In this paper, we proposed an efficient auditing scheme for outsourced big data such that DOs can verify data integrity without retrieving the data from the cloud storage. The proposed scheme utilizes algebraic signatures and an XOR-homomorphic technique, and enables the TTPA to check the integrity of large-scale data in the cloud. Analysis of security and performance shows that the proposed basic scheme is secure and efficient. Moreover, we designed a new data structure named RTable to extend our basic scheme to support the auditing of dynamic data, such as data modification, insertion, appendency, and deletion operations. By comparing our scheme with similar schemes, we showed that the extended scheme is highly efficient. Future work will involve attempting to construct a secure data auditing scheme with deduplication, which can help reduce data redundancy and save storage space in the cloud.

Acknowledgements This work was partially supported by National Natural Science Foundation of China (Grant Nos. 61070164, 61272415), Natural Science Foundation of Guangdong Province, China (Grant No. S2012010008767), and Science and Technology Planning Project of Guangdong Province, China (Grant No. 2013B010401015). This work was also supported by the Zhuhai Top Discipline-Information Security.

References

1 Demchenko Y, Ngo C, de Laat C, et al. Big security for big data: addressing security challenges for the big data infrastructure. In: Proceedings of Secure Data Management, Trento, 2013. 76–94
2 Wang B Y, Li B C, Li H. Oruta: privacy-preserving public auditing for shared data in the cloud. IEEE Trans Cloud Comput, 2014, 2: 43–56
3 Wang C, Chow S S M, Wang Q, et al. Privacy-preserving public auditing for secure cloud storage. IEEE Trans Comput, 2013, 62: 362–375
4 Shacham H, Waters B. Compact proofs of retrievability. In: Proceedings of International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, 2008. 90–107
5 Wang Q, Wang C, Ren K, et al. Enabling public auditability and data dynamics for storage security in cloud computing. IEEE Trans Parall Distrib Syst, 2011, 22: 847–859
6 Chen L, Zhou S, Huang X, et al. Data dynamics for remote data possession checking in cloud storage. Comput Electrical Eng, 2013, 39: 2413–2424
7 Zhang J, Dong Q. Efficient ID-based public auditing for the outsourced data in cloud storage. Inf Sci, 2016, 343: 1–14
8 Li J, Zhang L, Liu J K, et al. Privacy-preserving public auditing protocol for low-performance end devices in cloud. IEEE Trans Inform Forensic Secur, 2016, 11: 2572–2583
9 Wang Z, Han Z, Liu J. Public verifiability for shared data in cloud storage with a defense against collusion attacks. Sci China Inf Sci, 2016, 59: 039101
10 Wang H, He D, Tang S. Identity-based proxy-oriented data uploading and remote data integrity checking in public cloud. IEEE Trans Inform Forensic Secur, 2016, 11: 1165–1176
11 Yu Y, Au M H, Ateniese G, et al. Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage. IEEE Trans Inform Forensic Secur, 2017, 12: 767–778
12 Zhang R, Ma H, Lu Y, et al. Provably secure cloud storage for mobile networks with less computation and smaller overhead. Sci China Inf Sci, 2017, 60: 122104
13 Sookhak M, Gani A, Khan M K, et al. Dynamic remote data auditing for securing big data storage in cloud computing. Inf Sci, 2017, 380: 101–116
14 Schwarz T S J, Miller E L. Store, forget, and check: Using algebraic signatures to check remotely administered storage. In: Proceedings of the 26th IEEE International Conference on Distributed Computing Systems, Lisboa, 2006. 12–21
15 Chen L. Using algebraic signatures to check data possession in cloud storage. Future Generation Comput Syst, 2013, 29: 1709–1715
16 Sookhak M, Akhunzada A, Gani A, et al. Towards dynamic remote data auditing in computational clouds. Sci World J, 2014, 2014: 269357
17 Luo Y C, Fu S J, Xu M, et al. Enable data dynamics for algebraic signatures based remote data possession checking in the cloud storage. China Commun, 2014, 11: 114–124
18 Litwin W, Schwarz T. Algebraic signatures for scalable distributed data structures. In: Proceedings of the 20th International Conference on Data Engineering, Boston, 2004. 412–423
19 Ateniese G, Burns R, Curtmola R, et al. Provable data possession at untrusted stores. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, Alexandria, 2007. 598–609
20 Juels A, Kaliski B S. PORs: proofs of retrievability for large files. In: Proceedings of ACM Conference on Computer and Communications Security, Alexandria, 2007. 584–597
21 Yu Y, Zhang Y, Ni J, et al. Remote data possession checking with enhanced security for cloud storage. Future Generation Comput Syst, 2015, 52: 77–85

22 Ateniese G, Pietro R D, Mancini L V, et al. Scalable and efficient provable data possession. In: Proceedings of the 4th International Conference on Security and Privacy in Communication Networks, Istanbul, 2008. 1–10
23 Erway C C, Papamanthou C, Tamassia R. Dynamic provable data possession. ACM Trans Inf Syst Secur, 2009, 17: 213–222
24 Cash D, Küpçü A, Wichs D. Dynamic proofs of retrievability via oblivious RAM. J Cryptol, 2017, 30: 22–57
25 Yang K, Jia X. An efficient and secure dynamic auditing protocol for data storage in cloud computing. IEEE Trans Parallel Distrib Syst, 2013, 24: 1717–1726
26 Shen J, Shen J, Chen X, et al. An efficient public auditing protocol with novel dynamic structure for cloud data. IEEE Trans Inform Forensic Secur, 2017, 12: 2402–2415
27 Thangavel M, Varalakshmi P, Preethi T, et al. A review on public auditing in cloud environment. In: Proceedings of Information Communication and Embedded Systems, Chennai, 2016. 1–6
28 Ateniese G, Burns R, Curtmola R, et al. Remote data checking using provable data possession. ACM Trans Inf Syst Secur, 2011, 14: 1165–1182
29 Ren S Q, Tan B H M, Sundaram S, et al. Secure searching on cloud storage enhanced by homomorphic indexing. Future Generation Comput Syst, 2016, 65: 102–110
30 Ade-Ibijola A O. A simulated enhancement of Fisher-Yates algorithm for shuffling in virtual card games using domain-specific data structures. Int J Comput Appl, 2012, 54: 24–28
31 Barsoum A, Hasan A. On verifying dynamic multiple data copies over cloud servers. IACR Cryptol Eprint Arch, 2011, 2011: 447–476
32 Barsoum A, Hasan A. Enabling dynamic data and indirect mutual trust for cloud computing storage systems. IEEE Trans Parall Distrib Syst, 2013, 24: 2375–2385