Efficient Privacy Preserving Data Audit in Cloud

Hai-Van Dang, Thai-Son Tran, Duc-Than Nguyen,
Thach V. Bui, and Dinh-Thuc Nguyen

Faculty of Information Technology, University of Science, VNU-HCM, Vietnam
{dhvan,ttson,bvthach,ndthuc}@fit.hcmus.edu.vn,
thannguyen.hcmus@gmail.com

Abstract. With the development of database-as-a-service (DaS), data in cloud is more interesting for researchers in both academia and commercial societies. Despite DaS's convenience, there exist many considerable problems which concern end users about data loss and malicious deletion. In order to avoid these cases, users can rely on data auditing, which means verifying the existence of data stored in cloud without any malicious changes. Data owner can perform data auditing by itself or hire a third-party auditor. Until now, there are two challenges of data auditing as the computation cost in case of self auditing and data privacy preservation in case of hiding an auditor. In this paper, we propose a solution for auditing by a third-party auditor to verify data integrity with efficient computation and data privacy preservation. Our solution is built upon cryptographic hash function and Chinese Theorem Remainder with the advantage in efficient computation in all three sides including data owner, cloud server, and auditor. In addition, the privacy preservation can be guaranteed by proving the third-party auditor learns nothing about user's data during auditing process.

Keywords: Data Audit, Database-as-a-service (DaS), Cryptographic hash function, Chinese Remainder Theorem.

1 Introduction

Cloud computing and data service are two fast developing applications now. And the fields of those researches are also reinforced by requirements of the booming of smart phone and smart devices such as home appliances and wearable devices. It is trivial to see that two of the most requirements are data storage service and data privacy preservation. In those two requirements, data audit is the most important research because its result can give a direct solution of two those above requirements. In data storage service or database as a service, server of the third party is often assumed to be semi-trusted, i.e. it is believed not to use or reveal users data without permission. However, it may not reveal unexpected data loss or corruption to users, or delete some rarely accessed data sections. In order to avoid such risks, users must trust and rely on system of data auditing. The data auditing means verifying the existence of data stored in servers of cloud storage provider without any malicious deletion or modification. Data owner can perform data auditing by itself or hire an auditor. Here two challenges of data auditing are the computation cost in case of self-auditing and data privacy preservation in case of hiring an auditor. Since a decision made by a third party auditor will be not affected by

H.A. Le Thi et al. (eds.), *Advanced Computational Methods for Knowledge Engineering*,
Advances in Intelligent Systems and Computing 358, DOI: 10.1007/978-3-319-17996-4_17

different interest or benefit of both data owner and cloud server, the solution of hiring an auditor seems to be fair for the two sides.

Previous works of auditing methods by third party auditors consist of private verification and public verification. In detail, private verification scheme by Juels and Kaliski [4] encrypts data file and then embeds random value blocks (called sentinels) randomly. It is believed that if the server deletes a substantial portion of the file, a number of sentinels are also deleted and the auditor can discover this. This approach focus on only static file. The scheme by Shacham and Waters [5] bases on Message Authentication Code. They split erasure code of a file into blocks and each block into sectors, and create authenticators for each block based on Message Authentication Code (MAC). Every time the auditor wants to verify data, it sends a set of indices of blocks and the server returns corresponding data blocks along with their aggregated authenticator. The auditor afterwards applies MAC to verify integrity of blocks using its secret key shared by the data owner. This scheme gains advantage of fast computation but its downside is the server's response length and disclosure about verified data to the auditor.

The first public verification scheme by Ateniese et al. [1] was based on public key cryptosystem RSA. With this scheme, data are protected against the auditor and the server's response length is fixed by parameters of RSA cryptosystem. The scheme's downside is computation cost due to modular exponentiation operations. Another public verification scheme by Shacham and Waters [5] applies bilinear map [2] to aggregate authenticators of data blocks into one in order to reduce the server's response length. Later, a scheme by Wang et al. [6] applies pseudorandom generator to prevent the auditor from recovering data content from the message sent by the server; therefore, it provides data privacy-preservation against the auditor. The scheme's extension for batch auditing is straightforward and other schemes can be extended in the same manner.

In summary, auditing methods by third party auditors seem to focus on one or more problems among: reduction the server's response length, reduction computation cost in the server and the auditor, reduction storage overhead in the server and the auditor and data privacy preservation against the auditor, static data file or dynamic data file support. Among them, our motivation consists of computation cost reduction, efficient storage, data privacy preservation for data audit by an auditor and dynamic data file or database support.

In this paper, we propose solution for auditing by an auditor that uses secret key shared between data owner and auditor based on cryptographic hash function and Chinese Remainder Theorem. Thanks to fixed size of output of hash function, extra storage overhead for auditing is reduced. Shared secret key and Chinese Remainder Theorem provide the auditor a way to verify data in privacy preservation manner. In addition, the proposed approach gains efficient computation thanks to its simple calculations.

2 The Three-Parties Auditing Model

2.1 System Model

The model consists of three factors:

- Cloud server (CS): server that provides cloud database-as-a-service (DaS). The server is semi-trusted. It is believed not to use or reveal users data without permission. However, the server may not reveal unexpected data loss or corruption. Besides, the server may possibly decide to delete some data sections that have been rarely accessed by users.
- Data owner (DO): user who use DaS to keep their data. User wants to detect if any data has been lost, deleted or modified maliciously so it hires a third-party to audit data. However, the user does not want its data to be revealed to the third-party.
- Third-party auditor (TPA): a third-party that is hired by data owner to audit its data without knowing them. Data owner may send some information to TPA. When TPA wants to audit data in cloud server, it sends query to cloud server. Cloud server then returns corresponding metadata or proof back so that TPA can rely on to audit if data has been kept intact or not.

The design criteria of the above 3-parties model is a trade-off of three following goals:

- Low storage overhead. The additional storage used for auditing should be as small as possible on both TPA and CS.
- Low communication. The communication cost required by auditing process should be as low as possible.
- Low computational complexity. The computation for TPA should be low.

3 Proposed Method

3.1 Description

In this section, we give detailed exposition of the proposed auditing method and an example to illustrate it. In addition, we discuss about parameter selection and purpose of steps in the method.

Our proposed auditing method consists of three phases as described in the figure below.

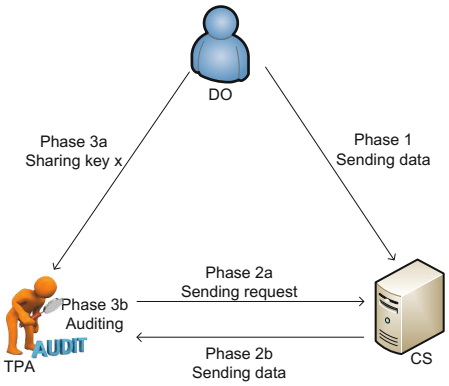

Fig. 1. The auditing process

Some conventions will be necessary for our description:

- x is shared secret key between the data owner and the auditor. Supposed that only data owner can upload data to server. the auditor cannot thanks to other access control policies of the server which we will not cover in this paper.
- $h : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^n$ is a keyed hash function that returns a value of n bits (n is a system parameter). $\{0,1\}^*$ is a binary string.
- $P : \{0,1\}^m \times \mathbb{N} \to \{0,1\}^m$ is a one-way permutation (this may be a uncompression hash function). $\mathbb{N}$ is the set of natural numbers and $\{0,1\}^m$ is a binary string of m bits.
- $Q : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^{n+1}$ is a prime generator that returns a $(n+1)$-bit prime.

Assuming that index of record in database is counted from 1.

Phase 1: The data owner (DO) sends data to the cloud server.

Each time the data owner sends a block of d data items. Assuming that the order of blocks is significant and counted from 0. Let us denote d records of the k^{th} block by $\{R_{kd+1}, \ldots, R_{(k+1)d}\}$. The data owner will

(1) Compute $S_k = \bigoplus_{i=1}^{d} P(R_{kd+i}, kd+i)$ where $\oplus$ is bitwise XOR operation.
(2) Compute $\overline{R_{kd+i}} = S_k \oplus P(R_{kd+i}, kd+i)\,(\forall i = 1, \ldots, d) = \bigoplus_{u=1,u\neq i}^{d} P(R_{kd+u}, kd+u)$.
(3) Compute $f_{kd+i} = h(\overline{R_{kd+i}}, x)\,(\forall i = 1, \ldots, d)$.
(4) Generate primes $q_i = Q(kd+i, x), (\forall i = 1, \ldots, d)$.
(5) Solve the system of congruences for the solution X_k

$$\begin{cases} X_k \equiv f_{kd+1} \pmod{q_1} \\ X_k \equiv f_{kd+2} \pmod{q_2} \\ \vdots \\ X_k \equiv f_{(k+1)d} \pmod{q_d} \end{cases} \tag{1}$$

(6) Send the k^{th} block and its metadata, $\{\{R_{kd+i}\}_{i=1}^{d}, X_k\}$, to the server.

Phase 2: The auditor sends its request to the cloud server, then the server sends back its proof

(1) The auditor sends to the server its request $\{I_j\}_{j=1}^{\ell}$ which are indices of records $R_{I_1}, \cdots, R_{I_\ell}, \forall j = 1, \cdots, \ell$. Given that $\{I_j = k_j d + i_j\}_{j=1}^{\ell}$, $1 \leq i_j \leq d$.
(2) For each index in the request, the server identifies the order of block containing the corresponding record $\{k_j = \lfloor \frac{I_j}{d} \rfloor\}_{j=1}^{\ell}$ and the order of the record in the identified block $\{i_j \equiv I_j \pmod{d}\}_{j=1}^{\ell}$. If $i_j \equiv I_j \pmod{d} = 0$ then assign $i_j = d$ (because the order or records in a block is counted from 1 to d)
(3) For each index I_j, the server computes $\overline{R_{I_j+1}} = \bigoplus_{u=1,u\neq i_j+1}^{d} P(R_{k_j d+u}, k_j d + u)$. If $i_j + 1 = d + 1$ then assign $i_j + 1 = 1$.
(4) The server returns $\{\overline{R_{I_j+1}}, X_{k_j}\}_{j=1}^{\ell}$ to the auditor.

Phase 3: The auditor verifies if the queried records are intact or not.

(1) For each index I_j of its request (given that $I_j = k_j d + i_j, 1 \leq i_j \leq d$), the auditor generates a prime $q_{i_j+1} = Q(I_j + 1, x)$ $(j = 1, \ldots, \ell)$.
(2) For each index I_j, the auditor compares whether

$$X_{k_j} \pmod{q_{i_j+1}} = h(\overline{R_{I_j+1}}, x)$$

is true or false. If true, all $d-1$ records of the block that contains R_{I_j}, except R_{I_j+1}, are ensured to be intact. If false, at least one record among those $d-1$ records has been being modified.

Discussion

Parameter Selection: For privacy preserving security, we select the size of block $d = 3$ (see section 4.2).
Purpose of Permutation Step: The permutation function P in our scheme helps to ensure to create different metadata $\overline{R}$ for the two same records R at different indices.
Purpose of the Hash Function h: It ensures to prevent finding back the record value R from the metadata $\overline{R}$.
Selection of the Number of Requested Records: According to A. Giuseppe et al. [3], if the server deletes t records and t is a fraction of n (the number of all records), for e.g. $t = 1\% \times n$, the auditor will detect this after challenging $c = 460$ records for auditing with probability of 95%. In our scheme, with a requested record, the auditor can verify $d-1$ records in a block. Therefore, with the same $c = 460$, the probability of successful detection is higher than 95%.

Next, an example will show how the audit method works.

Example 1. Without loss of generality, assuming that we will audit the first block (block of order $k = 0$) containing $d = 3$ records, $R_1 = 23, R_2 = 17, R_3 = 29$. Given secret key $x = 10$ and supposed that the three functions are:
$h(\overline{R_j}, x) = R_j \pmod{Q(j, x)}, \forall j = 1, .., 3$
$P(R_j, j) = R_j, \forall j = 1, .., 3$
$Q(1, x) = q_1 = 13, Q(2, x) = q_2 = 11, Q(3, x) = q_3 = 7.$
Phase 1:

(1) $S_0 = P(R_1) \oplus P(R_2) \oplus P(R_3) = R_1 \oplus R_2 \oplus R_3 = 23 \oplus 17 \oplus 29 = 10111_2 \oplus 10001_2 \oplus 11101_2 = 11011_2$.
(2) $\overline{R_1} = S_0 \oplus P(R_1) = S_0 \oplus R_1 = 11011_2 \oplus 10111_2 = 1100_2 = 12$
$\overline{R_2} = S_0 \oplus P(R_2) = S_0 \oplus R_2 = 11011_2 \oplus 10001_2 = 1010_2 = 10$
$\overline{R_3} = S_0 \oplus P(R_3) = S_0 \oplus R_3 = 11011_2 \oplus 11101_2 = 0110_2 = 6$
(3) $f_1 = h(\overline{R_1}, x) = \overline{R_1} \pmod{q_1} = 1100_2 = 12,$
$f_2 = h(\overline{R_2}, x) = \overline{R_2} \pmod{q_2} = 1010_2 = 10,$
$f_3 = h(\overline{R_3}, x) = \overline{R_3} \pmod{q_3} = 0110_2 = 6$
(4) Solve system of congruences

$$\begin{cases} X_0 \equiv f_1 \pmod{q_1} \\ X_0 \equiv f_2 \pmod{q_2} \\ X_0 \equiv f_3 \pmod{q_3} \end{cases} \Leftrightarrow \begin{cases} X_0 \equiv 12 \pmod{13} \\ X_0 \equiv 10 \pmod{11} \\ X_0 \equiv 6 \pmod{7} \end{cases}$$

We have $X_0 = 1000$

(5) Send $\{R_1 = 23, R_2 = 17, R_3 = 29, X_0 = 1000\}$ to the server.

Phase 2:

Assuming that R_2, a record in database, is modified to $R'_2 = 23 = 10111_2$. The server then computes $\overline{R'_3} = R_1 \oplus R'_2 = 10111_2 \oplus 10111_2 = 0$ and returns $\{\overline{R_3} = 0, X_0 = 1000\}$ to the auditor.

Phase 3:

The auditor can detect this change. Indeed, we have

With $X_0 = 1000, q_3 = 7$, $X_0 \pmod{q_3} = 6 \neq 0 = \overline{R'_3} \pmod{q_3}$.

It means all records the block containing the record R_2, except the record R_3, has been ensured to be intact.

3.2 Correctness

In order to prove correctness of the proposed audit method, at first we need to prove uniqueness property of solution of the system of congruences in Equation 1.

Theorem 1. *Let X be the solution of system of congruences that established in Phase 1 of proposed scheme (Equation 1):*

$$\begin{cases} X \equiv f_1 \pmod{q_1} \\ \vdots \\ X \equiv f_d \pmod{q_d} \end{cases}$$

then

(i) The system 1 has unique solution X in $\mathbb{Z}_{q_1 \times \cdots \times q_n}$ and
(ii) $X \pmod{q_i} = f_i (\forall i = 1, \ldots, d)$ and
(iii) $X \pmod{q_i} \neq g$ given that $g \neq f_i (\forall i = 1, \ldots, d)$.

Proof. (i) In Equation 1, $f_{kd+i} < q_i, \forall i = 1, \ldots, d$ because f_{kd+i} is a n-bit output of the keyed hash function h while q_i is $n+1$-prime output of prime generator Q. The primes $q_1, \ldots, q_d$ are different because they are output of the same generator Q with different input. This is a variation of the system in Chinese Remainder Theorem because $q_1, \ldots, q_n$ are distinguished primes. Therefore, it is correct due to Chinese Remainder Theorem.

(i) and (ii) It is clear.

Theorem 2 (Correctness). *If the server passes phase 3 of the proposed audit method, the corresponding blocks containing audited data must be kept intact in the server.*

Proof. For each index I_j of the request (given that $I_j = k_j d + i_j, 1 \leq i_j \leq d$), if the corresponding block of record R_{I_j} is kept intact, the server returns to the auditor $\{\overline{R_{I_j+1}}, X_{k_j}\}$ and X_{k_j} be the solution of the congruence system:

$$\begin{cases} X_{k_j} \equiv h\left(\overline{R_{k_j d+1}}, x\right) \pmod{q_1} \\ \vdots \\ X_{k_j} \equiv h\left(\overline{R_{k_j d+d}}, x\right) \pmod{q_d} \end{cases}$$

At the auditor side, by Theorem 1 (ii), it has $X_{k_j} \pmod{q_{i_j+1}} = h(\overline{R_{I_j+1}}, x)$, and the server passes phase 3.

If there is any record $R_{I_u}(I_u \neq I_j + 1)$ in this block that has been changed, then computed value $\overline{R_{I_j+1}}$ will be different from the computed value from the original data. This leads to change of the value $h(\overline{R_{I_j}+1}, x)$. Meanwhile, X_{k_j} is computed by the data owner based on the original data. Therefore, at the auditor side, $X_{k_j} \pmod{q_{i_j+1}} \neq h(\overline{R_{I_j+1}}, x)$. This is correct due to the Theorem 1 (iii).

4 Analysis

We will need some conventions in Table 1.

Table 1. Notations

Notation	Meaning
N	The number of records of database.
d	The number of records of a block.
R,$\|R\|$	R is a record. $\|R\|$ is the number of bits of the record R.
$\overline{R}$,$\|\overline{R}\|$	Combination of all records in the block except R. The combination is computed by bitwise XOR operation (same as step (2) in phase 1). It is easy to see that $\|\overline{R}\| = \|R\|$.
n	The number of bits of hash value created by the function $h : \{0,1,\}^* \times \{0,1\}^* \rightarrow \{0,1\}^n$
X	Metadata computed from a data block as in step (5) in phase (1). It is the solution of the system on d congruence equations in Equation 1 where each prime has $n+1$ bits. Therefore the number of bits of X is about $d \times (n+1)$.
$\|Int\|$	The number of bits of an integer
Xor^d	Take bitwise XOR operation of d records
$Prime_n^d$	Generating d primes of n-bit
$Perm_n^d$	Permute bits of d values of n-bit
$Hash_n^d$	Hash d values into d values of n-bit
$Solve_n^d$	Solve the solution for the system of d equations of congruences modulo n bit
$Modulo_n^\ell$	Compute ℓ modulo n-bit
$Compare_n^\ell$	Compare ℓ pairs of values of n-bit

4.1 Storage Overhead, Communication Cost and Computation Complexity

Due to the limit of pages, proofs for the following results are put in Appendix.

Theorem 3. *The storage overhead at the server side is $\mathcal{O}(N \times (|R| + n))$.*

Theorem 4. *1. The communication cost when the data owner sends a data block to the server is $\mathcal{O}(d \times (|R| + n))$.*

2. *The communication cost is $\mathcal{O}(\ell \times |Int|)$ from the auditor to the server and $\mathcal{O}(\ell \times (|R| + d \times n))$ from the server to the auditor when the auditor requests to audit ℓ records.*

Theorem 5. *1. The computation complexity at the data owner side is $Xor^d + Perm_n^d + Hash_n^d + Prime_n^d + Solve_n^d$ when the data owner sends one data block to the server.*

2. *The computation complexity at the auditor side is* $Prime_n^\ell + Modulo_n^\ell + Hash_n^\ell + Compare_n^\ell$ *when the auditor requests to audit* ℓ *records.*

4.2 Security

We concern the list of attack scenarios below:

1. Privacy preserving attack: The auditor recovers the content of record R from the proof $\overline{R}, X_k$ sent by the server
2. Replace attack: The server has deleted a record R. When the auditor requests to audit on the record R, the server selects another valid and uncorrupted record R' and its metadata to return to the auditor.
3. Replay attack: The server has deleted a record R_j. When the auditor requests to audit on the record R_j,the server generates proof from previous proof without querying the actual data.
   The proof of a record R consists of $\overline{R_{j+1}}, X$ where the size of $\overline{R_{j+1}}$ is the same as the size of R. In our context, the server is semi-trusted. Data loss is caused by unexpected corruption or by server's deletion to save storage. Therefore, the fact of deleting R_j but keeping its proof $\overline{R_{j+1}}$ does not make sense. We skip this kind of attack.
4. Forge attack: The server has deleted a record R_j. When the auditor requests to audit on the record R_j,the server forges its proof without querying the actual data.
   From details of phase 1 of the scheme, it is seen that the server needs to know the actual data and secret key x to create metadata. Therefore, we consider the secret key attack instead.
5. Secret key attack: The server finds the secret key shared between data owner and auditor.

Privacy-Preserving Attack. We have the following lemma.

Lemma 1. *Given* $a \oplus b, a \oplus c, b \oplus c$*, it cannot recover* a, b *or* c*.*

Proof. It can be proved by the following table

Table 2. Proof of lemma 1

ID	Given value	Equal expression
(1)	$a \oplus b$	(2) $\oplus$ (3)
(2)	$a \oplus c$	(1) $\oplus$ (3)
(3)	$b \oplus c$	(1) $\oplus$ (2)
(4)	0	(1) $\oplus$ (2) $\oplus$ (3)

We have $(1) = a \oplus b = (a \oplus b) \oplus (a \oplus c) = (2) \oplus (3)$. Similarly, $(2) = (1) \oplus (3), (3) = (1) \oplus (2)$. Therefore, from $(1), (2), (3)$, there is no way to find out a, b, c.

Based on the lemma 1, we will prove security for the proposed auditing method if the number of records in a block is $d = 3$.

Theorem 6. *From the server response* $\{\overline{R_{I_j+1}}, X_{k_j}\}_{j=1}^{\ell}$*, no information of records* $\{R_{I_j}\}_{j=1}^{\ell}$ *is leaked to the auditor if choosing* $d = 3$*.*

Proof. For each requested index I_j, given $I_j = k_j d + i_j (1 \leq i_j \leq d)$ and X_{k_j} is the solution of the system of congruences

$$\begin{cases} X_{k_j} \equiv h(\overline{R_{k_j d+1}}, x) \pmod{q_1} \\ \vdots \\ X_{k_j} \equiv h(\overline{R_{k_j d+d}}, x) \pmod{q_d} \end{cases}$$

where x is secret key, $q_i = Q(k_j d + i, x)$ is a prime,

$\overline{R_{k_j d+i}} = \bigoplus_{u=1,u\neq i}^{d} R_{k_j d+u}$ $(\forall i = 1, \ldots, d)$.

With $d = 3$, $\overline{R_{k_j d+1}} = R_{k_j d+2} \oplus R_{k_j d+3}$, $\overline{R_{k_j d+2}} = R_{k_j d+1} \oplus R_{k_j d+3}$, $\overline{R_{k_j d+3}} = R_{k_j d+1} \oplus R_{k_j d+2}$. According to Lemma 1, it cannot recover any record $R_{k_j d+1}$, $R_{k_j d+2}$, $R_{k_j d+3}$ in the block k_j from $\overline{R_{k_j d+1}}$,$\overline{R_{k_j d+2}}$, $\overline{R_{k_j d+3}}$.

Replace Attack Assuming that the auditor requests on the record R_I but the server returns proof $\overline{R_{J+1}}, X'$ for another record R_J. When the auditor verifies, it computes a prime $q = Q(I, x)$ and compares $X' \pmod{q} = h(\overline{R_{J+1}}, x)$. Actually $X' \pmod{q'} = h(\overline{R_{J+1}}, x)$ where $q' = Q(J, x)$ and $q \neq q'$. Therefore, $X' \pmod{q} \neq h(\overline{R_{J+1}}, x)$ and the server cannot pass the auditor's verifying.

Secret Key Attack

Theorem 7. *Supposed that the cryptographic hash function* h *is secure. From data sent from the data owner,* $\{R_{kd+i}\}_{i=1}^{d}, X_k$*, no information of the secret key* x *is leaked to the server.*

Proof. X_k is the solution of the system of congruences in Equation 1. Knowing X_k but $q_1, \ldots, q_d$, the server cannot compute $f_{kd+1}, \ldots, f_{kd+d}$. Note that even the server knows $f_{kd+1}, \ldots, f_{kd+d}$, it cannot recover the secret key x because $f_{kd+i} = h(\overline{R_{kd+i}}, x)$ where h is supposed to be secure.

4.3 Comparison to the State-of-Art Schemes

Compared to the two methods of private verification and public verification in the state-of-art [5], our proposed scheme has more efficient computation and storage overhead but heavier communication cost. Our efficient computation is thanks to simpler calculations of hashing and modulo compared to exponential and pairing [5]. Moreover, our storage overhead is proportional to the fixed size of hash function output instead of size of records as in [5], therefore it is more efficient. However, our proposed scheme has heavier communication since it can detect the exact block that is not intact and preserve data privacy against the third party auditor. Details of comparison is in the Appendix.

In addition, both state-of-art schemes and the proposed scheme can be applied for dynamic data file. This proposed scheme can be also applied for database. Basic operations (deletion, insertion, modification) are described in Appendix.

5 Conclusion

In this paper, we proposed an efficient and secure method of data audit. We then proved that our data auditing is privacy preserving based on cryptographic hash function and CRT, which can work online in cloud sufficiently. This can detect malicious deletion or modification with low computation cost. However, it will make an offset of communication cost to identify exactly the modified block while keeping data privacy. We believe that this is the future work of big data area with promising results.

References

1. Ateniese, G., Burns, R., Curtmola, R., Herring, J., Khan, O., Kissner, L., Peterson, Z., Song, D.: Remote data checking using provable data possession. ACM Transactions on Information and System Security (TISSEC) 14(1), 12 (2011)
2. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
3. Giuseppe, A., Randal, B., Reza, C., et al.: Provable data possession at untrusted stores. Proceedings of CCS 10, 598–609 (2007)
4. Juels, A., Kaliski Jr., B.S.: Pors: Proofs of retrievability for large files. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, pp. 584–597. ACM (2007)
5. Shacham, H., Waters, B.: Compact proofs of retrievability. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 90–107. Springer, Heidelberg (2008)
6. Wang, C., Wang, Q., Ren, K., Lou, W.: Privacy-preserving public auditing for data storage security in cloud computing. In: 2010 Proceedings IEEE INFOCOM, pp. 1–9. IEEE (2010)

A Appendix

A.1 Proof for Storage Overhead, Communication Cost and Computation Complexity

Theorem 8. *The storage overhead at the server side is* $\mathcal{O}(N \times (|R| + n))$.

Proof. Following phase 1 of the proposed auditing method in section 3.1, for each data block, a metadata is created and sent from the data owner. The number of blocks for the whole database is $\frac{N}{d}$. The size of a metadata is $d \times (n+1)$. Therefore, storage overhead at the server side is computed $N \times |R| + \frac{N}{d} \times d \times (n+1) = N \times (|R| + n + 1)$ or $\mathcal{O}(N \times (|R| + n))$

Theorem 9. *1. The communication cost when the data owner sends a data block to the server is* $\mathcal{O}(d \times (|R| + n))$.

2. *The communication cost is $\mathcal{O}(\ell \times |Int|)$ from the auditor to the server and $\mathcal{O}(\ell \times (|R| + d \times n))$ from the server to the auditor when the auditor requests to audit ℓ records.*

Proof. 1. Each time the owner sends a block of d records and its metadata to the server. Each record has size of $|R|$ bits and each metadata has size of $d \times (n+1)$ bits. Total cost of the data block and its metadata is

$$\text{Cost} = d \times |R| + d \times (n+1) = d \times (|R| + n + 1) \text{ or } \mathcal{O}(d \times (|R| + n))$$

2. When the auditor needs to audit data in the server, it sends a set of ℓ indices $\{I_j\}_{j=1}^{\ell}$ to the server. Each index is an integer. Therefore, total cost is $\mathcal{O}(\ell \times |Int|)$.
After receiving request $\{I_j\}_{j=1}^{\ell}$ from the auditor, the server sends back $\{\overline{R_{I_j+1}}, X_{k_j}\}_{j=1}^{\ell}$ where $|\overline{R_{I_j+1}}| = |R|$ bits and X_{k_j} has size of $d \times (n+1)$ bits. Therefore, communication cost is

$$\text{Cost} = \ell \times (2|R| + d \times (n+1)) \text{ or } \mathcal{O}(\ell \times (|R| + d \times n)).$$

A.2 Comparison with the State-of-Art Schemes

Denote that

- N is the number of records
- $|R|$ is size of a record
- n is size of output of hash function
- d is the number of record of a block
- $Mult^d$ is d multiplications
- $Rand^d$ is d random generating operations
- $Hash^d$ is d hashing operations
- Xor^d is d Xor operations
- $Prime^d$ is d prime generation operations
- $Solve^d$ is solving a system of d congruences

Comparison about computation cost, communication cost and storage overhead are presented in Table 3.

A.3 Database Operations

The three basic operations for database consists of data insertion, deletion and modification. Here we describe details of each operation so that our auditing method can be applied for database.

- Insertion: Each time the data owner sends a block of d records and the server always inserts a block at the end of the database. In case of less than d records, the data owner can wait until there are enough or create a few fake constant records.

Table 3. Comparison between the proposed scheme and the state-of-art schemes

	Private verification [5]	Pubic verification [5]	The proposed scheme
Keys	Secret key	A pair of public and private key	Secret shared key
Who can audit data?	Data owner	Any third party	Third party who is shared key with data owner
Aggregated proof	Yes	Yes	No
Computation cost to send 1 block (at data owner)	$Mult^d + Rand^d$	$Hash^d + Exp^d + Mult^d$	$Xor^d + Hash^d + Prime^d + Solve^d$
Computation cost of auditing ℓ records (at auditor)	$Mult^\ell$	$Hash^\ell + Exp^\ell + Mult^\ell + Pairing^2$	$Prime^\ell + Modulo^\ell + Hash^\ell$
Storage cost (at server)	$2N \times (\|R\|)$	$2N \times (\|R\|)$	$N \times (\|R\| + n)$
Communication cost (from server to auditor) when auditing ℓ records	$\|R\|d$	$\|R\|d$	$\ell(\|R\| + dn)$

- Deletion: If the data owner wants to delete a record R_i, he/she sends query to the server. The server firstly identifies the block containing R_i. Assuming that there are D blocks in the database and the identified block is the k^{th} block. If the last block has full of d records, he/she permutes the k^{th} block and D^{th} block, then sends the new last block to data owner for deleting record R_i and updating X_D before sending back to server. Otherwise, he/she sends the k^{th} and D^{th} blocks to data owner. The data owner then merges data of the two blocks so that the k^{th} block is full of d records. Data owner computes X_k and X_D. Finally he/she resends the updated k^{th} block with X_k and D^{th} block with X_D to server.
- Modification: If the data owner wants to modify a record R_i into R_i', he/she sends query to the server. The server firstly identifies which block contains R_i. Assuming that it is k^{th} block. It returns the k^{th} block to the data owner. The data owner modifies R_i and recalculates X_k before sending back to server.