Improving efficiency of remote data audit for cloud storage

Kuan Fan[1], Mingxi Liu[2] and Wenbo Shi[1*]

[1] School of computer and Communication Engineering, Northeastern University
Qinhuangdao 066000, Hebei - P.R. China

[2]School of computer and Communication Engineering, Northeastern University
Shenyang 110819, Liaoning - P.R. China

*Corresponding author: Wenbo Shi
[e-mail: swb319@hotmail.com]

Received February 6, 2018; revised September 5, 2018; accepted November 13, 2018; published April 30, 2019

Abstract

The cloud storage service becomes a rising trend based on the cloud computing, which promotes the remote data integrity auditing a hot topic. Some research can audit the integrity and correctness of user data and solve the problem of user privacy leakage. However, these schemes cannot use fewer data blocks to achieve better auditing results. In this paper ,we figure out that the random sampling used in most auditing schemes is not well apply to the problem of cloud service provider (CSP) deleting the data that users rarely use, and we adopt the probability proportionate to size sampling (PPS) to handle such situation. A new scheme named improving audit efficiency of remote data for cloud storage is designed. The proposed scheme supports the public auditing with fewer data blocks and constrains the server's malicious behavior to extend the auditing cycle. Compared with the relevant schemes, the experimental results show that the proposed scheme is more effective.

Keywords: Cloud storage, PPS, public auditing, auditing result statistics

1 . Introduction

Cloud computing is a type of distributed computing that automatically splits a large number of computational processes into numerous smaller subprograms through a network and sends them to a huge system of multiple servers for searching. After calculating and analyzing, the processing results are transmitted back to the user. In 2011 the National Institute of Standards and Technology(NIST) defines that cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [1]. In 2015, the typical cloud service market represented by IaaS, PaaS and SaaS reached 52 billion 240 million US dollars, and it will be expected to 143 billion 530 million US dollars by 2020 [2].

Although cloud computing has many advantages, there are still some debates and hesitations before it is widely used. Data security and privacy are the major considerations when users use cloud computing [3]. Firstly, the users store personal data on the cloud server and then delete the local copy, which means that they lose control of their own data. The cloud server may intentionally delete part of the users' data for his own financial benefit [4,5]. Secondly, because of server hardware damage and some irresistible reasons, the user's data will be damaged. When the problems above occur, the cloud server may try to hide these errors, and make the users believe that the data is still stored on the cloud server correctly. Thirdly, when the third part auditor (TPA) is interested in certain data of a user, he may constantly challenge these data. The TPA obtain the data through calculating the proof returned by CSP, which can lead to privacy leakage of user data [6]. Thus, how to efficiently verify correctness of outsourced data without the local copy and protect the privacy of users data become a big challenge for data storage security in cloud computing. In order to solve the challenge, many studies present different schemes and security models [7-18], these schemes improve the data audit continuously, and strive to achieve the goal of universal, high security, strong privacy and efficient.

In 2007 Ateniese et al. proposed a provable data possession (PDP) model [7]. The PDP protocol supports public auditability to ensure that data is securely stored on untrusted servers, but it only supports static auditing and does not support dynamic data verification. In 2008, Ateniese et al. proposed another scalable PDP protocol that supports dynamic auditability but cannot support full dynamic, such as insert operations [8]. The next year, Wang et al. proposed the challenge-response protocol, which can detect the correctness of the data and the location of the errors, but it still cannot support full dynamic verification [9]. In the same year, Erway et al. proposed a dynamic provable data possession that extends the PDP model to support fully dynamic data [10]. However, public auditing is not supported. In

2007, Juels and Kaliski proposed the proof of retrievability protocol (POR) that allows data recovery, which not only checks data integrity but also restores the original data [11]. However, their scheme only supports static data storage. The next year, Shacham and Waters proposed an improved POR protocol that uses BLS signatures instead of RSA signatures to shorten the length of evidence in auditing, but the protocol also considers only static data [12]. In 2011, Wang et al. proposed a new protocol to solve the problem of full dynamic public auditing based on the above agreement [13]. In 2013, Wang et al. [14] pointed out that the Wang et al.'s [13] protocol had privacy issues. The TPA can calculate user data through audit proof and Wang et al. proposed the scheme to avoid this privacy issue. As a result, the dynamic public verification agreement was basically achieved. Since then, several researchers have studied the factors of auditing efficiency, auditing background and user key security based on the above schemes [15-19]. For example, in the dynamic auditing process, the audit data block size is fixed, which affects the efficiency of updating. In 2014, Liu et al. proposed a protocol that supports audit data blocks are not fixed in size during dynamic auditing [20]. In 2017, Song et al. proposed a signature mechanism with additive homomorphic operations to handle the modification of shared data on the server in the process of dynamic auditing [21]. This signature mechanism supports correctness and completeness verification of multi-user modifications without requiring an always-online data owner. At the same year, Fu et al. proposed a new privacy-aware public auditing mechanism to address the issue of public auditing on the integrity of shared data that may reveal data owners' sensitive information [22]. The Fu et al.'s protocol designed that t managers of member groups cloud jointly generate a trace key to prevent abuse of audit rights and construct a binary tree structure. Group members could trace the data changes and roll back the data. In 2017, Yu et al. proposed an anti-key compromise protocol to solve the problem of user key leakage in the auditing process [23]. At different stages of the Yu's protocol, users generated different key signatures. As long as the attacker cannot obtain the current key, the attacker does not pose a threat to the security of the data. In 2017, Wang et al. put forward the incentive and unconditionally anonymous identity-based public PDP (IAID-PDP) protocol to solve the problem of user identity privacy in the auditing process [24]. The Wang et al.'s scheme treats agencies as a judiciary, providing identity protection and incentives to users who provide important information. At present, most of audit schemes mainly focus on the auditing performance, auditing role changes, user privacy and other aspects based on the classic audit protocols. However, few researchers focus on selecting audit data to improve audit efficiency. Our scheme puts attention to this issue.

In the majority of audit schemes, the random method is used to extract challenge data blocks. No one cares about how to choose data to improve audit efficiency. Is random method the most efficient way of selecting data? It is generally known that data is accessed with a certain frequency and cycle. For this feature, the proposed scheme uses the PPS method to select challenge data blocks. The PPS refers to the probability sampling that the

population is divided into primary sampling units (PSU) of unequal capacity based on the auxiliary information [25]. In multi stage sampling, especially two stage sampling, the probability of sampling a PSU depends on the size of PSU in the PPS. The larger the PSU, the greater the probability of the PSU being selected. The smaller the size of the PSU, the less the probability of the PSU being drawn. For this reason, the PPS has a wide range of applications [26-28]. In the proposed scheme, the cloud server will delete the data that is not commonly used by users, so the frequency of data is chosen as ancillary information for dividing users data to form PSU. Therefore, PSU consisting of uncommonly used data has a large scale, and the probability of picking such PSU as challenge data blocks is very high. The experimental results show that the PPS-selecting applied by our scheme is more efficient than random-selecting in general audit scheme. It can generate the same auditing result as the random method with fewer challenge blocks.

In public auditing, after the TPA completes the audit, it will store a lot of auditing results about users data. At present, few articles consider how to use these results to feedback for improving audit scheme's efficiency. In the proposed scheme, TPA is an honest party that can collect the auditing results. The TPA may have a certain judgment on the cloud server's credit according to auditing results after a period of time. Announcing these auditing results can affect the reputation of the cloud server, which forces the server to improve the data integrity and give users enough confidence to increase or decrease their auditing cycle. Specifically, the contribution in this work can be summarized as the following three aspects:

1) CSP deletes the data that user infrequency used for the economic or benefit reasons [6,7,29,30]. The PPS is used to extract data blocks as challenge data blocks. Under the same auditing conditions, the PPS has a higher accuracy than the common random sampling to identify the malicious behavior of CSP.
2) TPA collects auditing results of users who use CSP storage server. After a period of time, TPA can estimate the credit of CSP based on the users' auditing results, which can force CSP to improve service quality.
3) In order to illustrate effectiveness and security of the proposed scheme, security analysis and experimental comparison of the proposed scheme show that the scheme is indeed safe and effective.

The rest of this paper is organized as follows: Section 2 introduce some definitions. Section 3 presents the system model and the design goals. The detailed description of the proposed scheme are introduced in Section 4. Section 5 gives the proposed scheme provide the security analysis of the proposed scheme. The evaluation of performance is shown in Section 6. Finally, the concluding remark of the whole paper is given in Section 7.

2 . Preliminaries

2.1 Bilinear maps

Let G_1, G_2 be two multiplicative cyclic groups of prime order p, and g be a generator of G_1. A bilinear map $e: G_1 \times G_1 \to G_2$ with the following properties [31]:

1) *Computability*: there exists an efficiently computable algorithm for computing map $e: G_1 \times G_1 \to G_2$

2) *Bilinearity*: for all $u, v \in G_1$ and $a, b \in Z_p^*$, $e(u^a, v^b) = e(u, v)^{ab}$

3) *Non-degeneracy*: $e(g, g) \neq 1$

2.2 The PPS

The PPS is belong to Unequal Probability Sampling [25]. The PPS is described as follows [32]: Let there be N first-stage sampling units, $C_1, C_2, ..., C_n$ of sizes $M_1, M_2, ..., M_t, ..., M_N$, respectively. n first-stage sampling units are drawn from N units. Further, let $(i_1, i_2, ..., i_n)$ denote any combination of n integers taken from N integers, $1, 2, ..., N$. In drawing n first-stage sampling units, we employ the probability proportionate to the sum of their sizes, that is, $C_{a1}, C_{a2}, ..., C_{an}$ is drawn with the probability proportional to

$$\sum_{i=1}^{n} M_{ai} = M_{a1} + M_{a2} + ... + M_{an} \tag{1}$$

$$\Pr(C_{a1}, C_{a2}, ..., C_{an}) = \frac{\sum_{i=1}^{n} M_{ai}}{\binom{N-1}{n-1} M} \tag{2}$$

Where $M = \sum_{i=1}^{M} M_i$

For the second-stage sampling, l second-stage sampling units are subsampled employing the probability which is equal for any combination of l second-stage sampling units in C_a. The specific PPS algorithm is as follows:

As described above, the total number of units is N, n units are drawn from N in the first-stage sample. The value of interval sampling is $K = \frac{M}{n}$. A value R is extracted randomly between $1 \sim K$, so the corresponding unit containing R is the extracted unit, then the units at every K size measurement are drawn (e.g., $R+K, R+2K, R+3K, ..., R+(n-1)K$).

3 . Problem statement

3.1 System model

The proposed scheme considers a cloud data storage service involving four different entities, as illustrated in **Fig. 1**: the cloud users who have large amount of data that need to store in the cloud, and these users are independent of each other; the CSP who is the cloud server administrator, has certain amount of storage space and computing resources to provide the users' data; the TPA, which is a third party and has a certain computation resources and communication capability, is trusted by users can interact with CSP to audit user data [30]; the public which are using storage services or want to use storage services provided by the CSP.

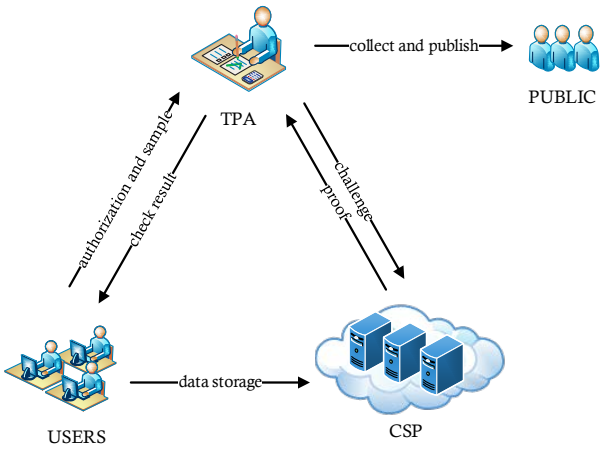

Fig. 1. The system model

In order to improve the auditability efficiency, user uses the PPS method to choose challenge data blocks. Therefore, in the preprocessing stage, user sorts the data by frequency and then generates a dataset which contains data frequency and id information for PPS. The data frequency is chosen as auxiliary information to divide the outsourced data into a number

of PSU. Meanwhile, user generates homomorphic verifiable tags and sends these tag and the outsourced data blocks to the CSP. When the user initiates an audit request, he uses the PPS to select a number of challenge data blocks' id sent to the TPA along with audit authorization. Then, the TPA sends an auditing challenge together with the audit authorization to the CSP. After receiving these messages, the CSP checks the legitimacy of the TPA's authorization. If valid, the CSP responds to the TPA with a proof; otherwise, is not. Finally, The TPA checks the correctness of the proof and sends an auditing result to user, then it stores the auditing result. In order to reduce the calculation pressure of the auditing system, the TPA counts the auditing results and publishes them regularly, which can constrain the CSP malicious behavior and help the public make some decisions such as selecting storage servers and determining auditing cycles.

3.2 Threat model

In this work, assuming CSP is "semi-honest" [7,33]. That means CSP will forge proof when the outsourced data to be audited is destroyed. The TPA considered to be "honest-but-curious" [33]. That is to say, it audits the data correctly, collects and publishes the auditing results honestly, but it is curious about users data. There are two types of attackers: 1) An internal attacker refers to the CSP; 2) A privacy attacker refers to the TPA. In this paper, three potential security threats are consider:

1) **Data corruption:** The adversaries, for their own benefit, might neglect to keep or deliberately delete infrequently accessed data which are owned by cloud users. Their goal is corrupting users data without being checked by TPA. The adversaries could be the CSP or other internal attacker.
2) **Forgery attack:** The adversaries may destroy some data from a user and forge these data tags without knowing the user's private key for maintaining their reputation. The adversaries could be the CSP or other internal attacker.
3) **Privacy disclosure:** The adversaries may infer some users data from the proof during the auditing process. The adversaries aim at getting users data without informing users. The adversaries could be the TPA or other privacy attacker.

3.3 Design goals

To efficiently check the integrity of user data, the proposed scheme should be designed to achieve the following properties:

1) **Public auditability:** to allow that the TPA verify the correctness of users data storage on the cloud without a copy of the whole data, which reduce the users' computational burden.
2) **Storage correctness:** to ensure that the CSP that does not store users' data cannot pass the auditing.

3) **Privacy-preserving:** to ensure that there is no way for TPA to infer users' data content from the proofs during the auditing process.
4) **Effectiveness:** to ensure that the PPS method can detect malicious behavior of CSP using fewer data blocks than random method. The TPA is allowed to publish the statistical data calculated from auditing results for reducing the CSP's malicious behavior.

4 . The proposed scheme.

4.1 Description of the Scheme

Fig. 2 is the algorithm flow chart of the scheme, the detailed algorithm is as follows:

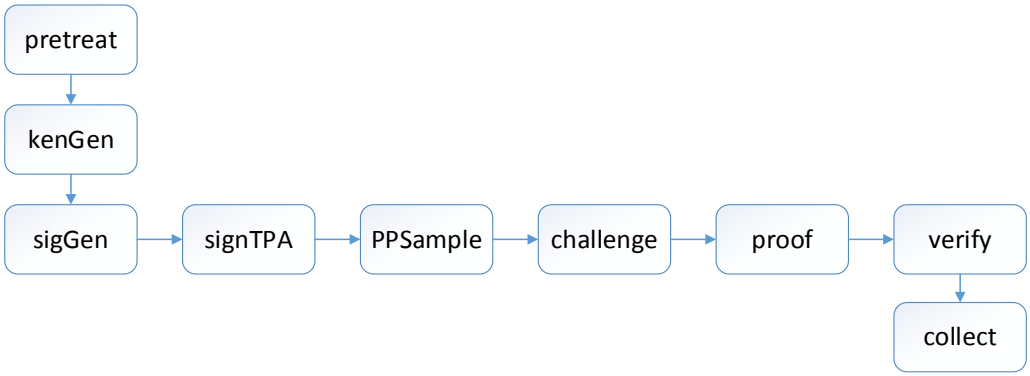

Fig. 2. Algorithm flow chart

1) Initialization parameters

Let G_1 and G_2 be two multiplicative cyclic groups of prime order p .The global security parameters of the proposed scheme are $(G_1, G_2, e, p, g, u, H)$,where the $e: G_1 \times G_1 \to G_2$ be a bilinear map as introduced in preliminaries, g be a random generator of G_1, u be a random element of G_1 ,and $H: \{0,1\}^* \to G_1$ is a secure one-way hash function from the arbitrary strings to the elements in G_1.

2) Algorithm $pretreat()$:

The pretreatment concludes two algorithms. The first algorithm generates a new dataset M_{local}, that is keep locally for PPS .The M_{local} consists of the frequency and id of user data block. The input is $M=\{m_1,m_2,...,m_n\}$ that will be uploaded to the CSP by user; the output is M_{local}. The algorithm is as follow:

Algorithm 1 the dataset for PPS

Input: $M=\{m_1,m_2,...,m_n\}$

Output: $M_{local}=\{m_1^{'},m_2^{'},...,m_n^{'}\}$

1. $read(M)$
2. $M_o \leftarrow sorted\ M\ by\ M.frequency$
3. dim M_{local} *as a list*
4. *for* $i=1$ *to n do*:
5. dim *dict as a dictionary*;
6. $dict["frequency"] \leftarrow M_o[i].frequency$
7. $dict["id"] \leftarrow M_o[i].id$
8. *add dict at the end of* M_{local}
10. E*nd for*
11. *return* M_{local}

The second algorithm generates a cumulative Table T_{cum} for PPS. The frequency is sampled in this cumulative table using PPS. The input are the Minimum frequency $f_{\min}$, the Maximum frequency $f_{\max}$ and the frequency interval interv; the output is T_{cum}. The algorithm is as follow:

Algorithm 2 cumulative Table for PPS

Input: $f_{\max}$, $f_{\min}$, interv

Output: T_{cum}

1. dim T_{cum} *as a list*
2. $rangeLow = 1, sum = 0$
2. *for* $i \leftarrow f_{\min}$ *to* $f_{\max}$ *do*:
3. dim *temp as a list*
4. $freqInterv \leftarrow i + \text{int}erv$
5. *if* $freqInterv \geq f_{\max}$ *then*:
6. $freqInterv \leftarrow f_{\max}$
7. *End if*
8. *frequency* $\leftarrow i$, *freqInterv*
9. *while*($i \leq freqInterv$) *do*:
10. $sum \leftarrow sum + i$;
11. $i \leftarrow i + 1$;
12. *End while*
13. $range \leftarrow rangeLow, sum$
14. $rangeLow \leftarrow sum + 1$
15. *add frequency*, *sum*, *range to the end of temp*
16. *add temp to the end of* T_{cum}
17. *End for*
18. T_{cum} *is written to CSV*, *column name is frequency, sum, range, respectively*

The table T_{cum} generated by algorithm 2 is as follow:

Table 1. Cumulative Table T_{cum}

id	frequency	sum	range
1	$f_1 \sim f_{\text{interv}}$	$sum_1 = \sum_{i=1}^{\text{interv}} f_i$	$f_1 \sim sum_1$
2	$f_{\text{interv}} + 1 \sim f_{2*\text{interv}}$	$sum_2 = \sum_{i=f_1}^{2*\text{interv}} f_i$	$sum_1 + 1 \sim sum_2$
…	…	…	…
n	$f_{(n-1)*\text{interv}} + 1 \sim f_{\max}$	$sum_n = \sum_{i=f_1}^{f_{\max}} f_i$	$sum_{n-1} + 1 \sim sum_n$

3) Algorithm $keyGen(1^k)$:

The user first chooses a random value $\alpha \leftarrow Z_P$ as his private key sk_u, then he computes $v \leftarrow g^{\alpha}$ as his public key pk_u. After that, the user uploads his public key pk_u to the TPA and stores his private key sk_u locally. The CSP generates his key pair $\{sk_{csp}, pk_{csp}\}$, he sends

his public key pk_{csp} to the TPA, and keeps his private key sk_{csp} locally.

4) Algorithm $sigGen(M, sk_u)$:

For each $m_i \in M (i \in [1,n])$, user computes signature $\sigma_i = (H(m_i).u^{m_i})^{\alpha}$ by his private key α and collects a signature set $\Phi = \{\sigma_i\}_{1 \le i \le N}$. The user generates the tag $tag = name \| \mathrm{n} \| u \| sig_{sk_u}(name \| \mathrm{n} \| u)$ of M, where $name \in Z_P^*$ is a random value which is chosen as the identifier of M by the user. The user sends $\{M, \Phi, tag\}$ to the CSP and deletes M and Φ from his local storage.

5) Algorithm $signTPA(sk_u, tag)$:

The user asks the TPA for his *ID* in order to grant it audit permission. The TPA return *PID* that is the ciphertext of his *ID* using the CSP's public key. The user computes $sig_{TPA} = sig_{sk_u}(AUTH \| PID \| tag)$, where the *AUTH* is a random value selected by the user.

Then, the user sends *AUTH* to the CSP and sig_{TPA} to the TPA.

6) Algorithm $PPSample(M_{local}, T_{cum}, n, m)$:

The proposed scheme samples *n* units using PPS from the cumulative Table T_{cum} in the first stage. Normally, *n* units have lower frequency. In the second stage, *m* elements are drawn from every unit using random sampling. Therefore, a total of $n*m$ elements are sampled. The input are M_{local}, T_{cum}, n and *m*. The output is a dataset *IChal* that contains the id of $n*m$ elements. The algorithm is as follow:

Algorithm 3 PPSample

Input: M_{local}, T_{cum}, n, m

Output: *IChal*

1. *read* T_{cum}
2. *read* M_{local}
2. $sumMax \leftarrow$ *the Maximum sum value in the* T_{cum}
3. $k \leftarrow sumMax / n;\ j \leftarrow 0;$
4. $R \leftarrow$ *the random value between* $[1, k]$
5. dim L, Fre *as list*
6. *for* $i = 1$ *to n do*: /**the first sampling stage**/
7. *add k to the end of L*
8. $k \leftarrow k + R$
9. *End for*
10. *for t in* T_{cum}
12. *while*($L[j]$ *is between t.range*) *do*:
13. *add t.frqeuency to the end of Fre* /**sampling results**/
15. *End while*
16. E*nd for*
17. *for m in* M_{new} *do*: /**the* sec*ond sampling stage**/
18. *if m.frequency is between* $Fre[z]$ *then*:
19. *add m.id to list G*
20. *else m.frequency* $>$ $Fre[z]$*.lastfrequency*
21. z $\leftarrow$ z+1
22. *Randomly select m elements from G and add them to IChal*
23. *empty G*
24. *End if*
25. *End for*
26. *rentuen IChal*

7) Algorithm $challenge(pk_{csp}, sig_{TPA}, IChal)$:

According to the audit requirements in the public auditing process, TPA chooses the random coefficient $v_i \leftarrow Z_p$ where $i \in IChal$, and then it generates the challenge message $chal = \{sig_{TPA}, \{PID\}_{pk_{csp}}, \{i, v_i\}_{i \in IChal}\}$. The TPA sends the challenge message to the CSP.

8) Algorithm $proof(chal, \Phi, M)$:

Upon receiving the challenge message $chal$, the CSP decrypts $\{PID\}_{PK_{csp}}$ with its private key sk_{csp}, then he uses $AUTH$, PID, tag and user's public key pk_u to verify whether this TPA is indeed authorized by user through the following equation

$sig_{TPA} = e((H(AUTH \| PID \| tag))^{\alpha}, g) \stackrel{?}{=} e(H(AUTH \| PID \| tag), v)$. If the equation equals, the CSP computes $\mu = \sum_{i=I_1}^{IChal} v_i m_i \in Z_P$ and $\sigma = \prod_{i=I_1}^{IChal} \sigma_i^{v_i}$, where $IChal$ is drawn by the PPS. Then the CSP will respond to the TPA with a proof $P = \{\mu, \sigma, H(m_i)\}_{I_1 \le i \le IChal}$.

9) Algorithm $verify(pk_u, chal, P)$:

The TPA checks the received proof :

$$e(\sigma, g) \stackrel{?}{=} e(\prod_{i=I_1}^{I_c} H(m_i)^{v_i}.u^{\mu}, v) \tag{3}$$

If it equals, the algorithm returns TRUE, and the TPA believes that the data stored on the cloud is complete, otherwise, it returns FALSE. The TPA sends the result to the user.

10) Algorithm $collect(auditingResults)$:

The TPA collects the auditing results of each user using the CSP storage service and classifies auditing results according to the size of challenge data blocks. After a period of time, the TPA will get statistics about storage behavior of the CSP.

5 . Security analysis

The safety of the proposed scheme will be discussed from several aspects:

Theorem 1. When the user data are unfaithfully stored in the cloud, the CSP cannot generate valid proof P through the TPA audit.

Proof: Based on the properties of bilinear maps, the correctness of the proposed scheme can be proved by the verification equation through deducing the left hand side from the right hand side.

$$\begin{aligned} e(\sigma, g) &= e(\prod_{i=I_1}^{IChal} \sigma_i^{v_i}, g) \\ &= e(\prod_{i=I_1}^{IChal} (H(m_i).u^{m_i})^{\alpha})^{v_i}, g) \\ &= e(\prod_{i=I_1}^{IChal} (H(m_i).u^{m_i})^{v_i}, g^{\alpha}) \\ &= e(\prod_{i=I_1}^{IChal} (H(m_i)^{v_1}).u^{\sum_{i=I_1}^{IChal} m_i v_i}, v) \\ &= e(\prod_{i=I_1}^{IChal} (H(m_i)^{v_1}).u^{\mu}, v) \end{aligned}$$

Therefore, as long as the cloud server faithfully stores the user data, the audit equation will be valid.

Theorem 2. In the authorization process, the TPA cannot receive the CSP's response with an integrity proof P unless this TPA obtain the authorization from user.

Proof: The TPA cannot forge audit authorization sig_{TPA} by himself in the proposed scheme. The TPA don't know the private keys sk_u that generated the authorized signature. If the TPA forged the authorization, the CSP will judge it during the bilinear maps verification process using the user-generated public key pk_u. The CSP will not generate evidence as long as the TPA authorization is not validated.

Theorem 3. Though the process of selecting the challenge data blocks, PPS method is better than simple random sampling in dealing with the situation of CSP deletes the data that the user has not used for a long time.

Proof: The proof consists of two steps. Firstly, it is necessary to prove that the PPS effect is the same as the simple random sampling if the size of the group is the same or similar.

Assuming there are *n* pieces of data, and *m* pieces of data are extracted from *n*. The probability of simple random sample is $\frac{m}{n}$. In the PPS, the data is divided into a groups, each group has $\frac{n}{a}$ data. Two-stage sampling, in the first stage, extract *b* groups from a groups, and in the second stage $\frac{m}{b}$ data will be extracted from *b* groups. So the probability of extracting n pieces of data can be expressed as $p(\alpha\beta) = p(\alpha)p(\beta \mid \alpha)$, where $p(\alpha)$ is the probability of extract *b* groups from a, and $p(\beta \mid \alpha)$ is a conditional probability of extracting $\frac{m}{b}$ data in the second stage when the *b* group has been extracted in the first stage:

$$\begin{aligned} p(\alpha\beta) &= p(\alpha)p(\beta \mid \alpha) \\ &= \frac{b}{a} \times \frac{\frac{m}{b}}{\frac{n}{a}} \\ &= \frac{m}{n} \end{aligned}$$

Thus, if there is as much data as possible in each group, the probability of PPS drawing data is the same as simple random sampling.

Secondly, it is need to prove that PPS tends to extract large groups instead of small ones, if the group sizes are different. Assuming there are n pieces of data, and m pieces of data are extracted from n, the probability of sample is $p(\alpha\beta)=\frac{m}{n}$. The number of groups drawn in the first stage is fixed at b, then in the ideal state, no matter the size of the group, each group extracted from the c pieces of data. So $m=b\times c$ and $p(\beta|\alpha)=\frac{c}{k_i}$, where k_i indicates the size of the group. According to the formula $p(\alpha\beta)=p(\alpha)p(\beta|\alpha)$, calculate $p(\alpha)$:

$$\begin{aligned} p(\alpha) &= \frac{p(\alpha\beta)}{p(\beta|\alpha)} \\ &= \frac{m/n}{c/k_i} \\ &= \frac{b\times c/n}{c/k_i} \\ &= \frac{b\times k_i}{n} \end{aligned}$$

Thus, if a group have more data, then the probability of sampling data in this group is relatively large by the PPS method.

The first step of proof shows that the probability of PPS extracting data is the same as simple random sampling if the amount of data in each group is the same; the second step of proof means that PPS prefers to the group of more data in the sampling process under the condition of the same probability. So compared to simple random sampling, PPS prefers grouping with more data and extracts data from it.

6 . Performance evaluation

6.1 Experimental results

In order to evaluate the efficiency of the proposed scheme, the experiment is conducted on a Linux server with 2.7 GHz CPU and 4 GB memory on the Ubuntu system. In the experiment, the GNU Multiple Precision Arithmetic (GMP) [34] and Pairing-Based Cryptography (PBC) [35] library are used to achieve the proposed audit algorithm. The PPS algorithm is implemented by Python language and the rest are based on C language. The experiment employs a $MNT\ d59$ curve, which has a 160-bit group order. Thus, $|p|$ is 160 bits. The size of each data block is the same, 167 bits.

Pretreatment and authenticators generation: According to section 4.1, the user have to pre-process the data blocks, including grouping the data blocks by frequency and generating data signatures, and then sending the processed dataset to the CSP. In order to prove that the process of the pretreatment costs little extra computation overhead, the following experiment is designed. **Fig. 3** shows the time of grouping data blocks and generating authenticators with different number of data blocks. The time spend on the proposed scheme is compared with the SW scheme [12] that does not support data grouping by frequency. From **Fig. 3**, it can be seen that the time of our scheme is basically the same as the SW scheme. Processing 10,000 data blocks, our program spends 28.221 seconds, while the SW spent 27.8 seconds. The time difference between the proposed our scheme and the SW's scheme is 0.01 seconds to 0.42 seconds from the treatment of 1000 to 100000 data blocks.

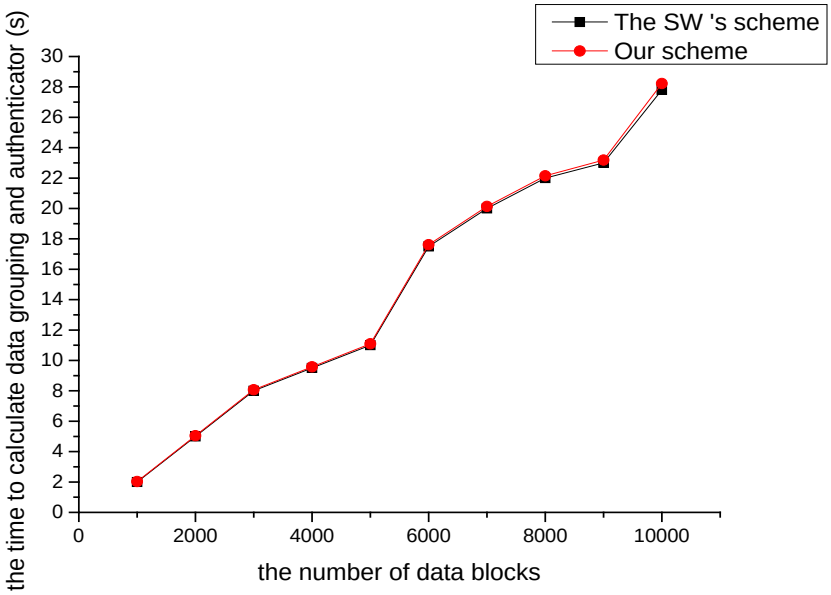

Fig. 3. Computation cost of grouping and generating authenticators

Performance of Auditing: As mentioned in Section 4.1, when users want to audit data stored on CPS, they use the PPS method to select a certain number of data ID from the dataset they keep and then authorize the TPA to prevent disclosure of user privacy. In order to evaluate the overhead of auditing computation in the proposed scheme, three diagrams are illustrated, which are sample time, proof generation time and audit proof time. **Fig. 4** shows that the time of of extracting 100-1000 data blocks from 1 million data. The time spend on the PPS is compared with the random sample that is used in most auditing schemes. Because the PPS has two sampling processes, it spend more time than random sample. The time difference between the PPS and the random sample is 0.19 seconds to 0.2 seconds from the treatment of 100 to 1000 data blocks. **Fig. 5** manifest the time spend on generating proof with different number of data blocks. It compared with the SW's scheme. It can be seen that

our scheme's time is close to the SW's. **Fig. 6** shows the time of verify proof. From **Fig. 6**, it can be seen that the time of our scheme is basically the same as the SW's scheme. The time difference between the our scheme and the SW's scheme is 0.001 seconds to 0.05 seconds from the treatment of 100 to 1000 data blocks.

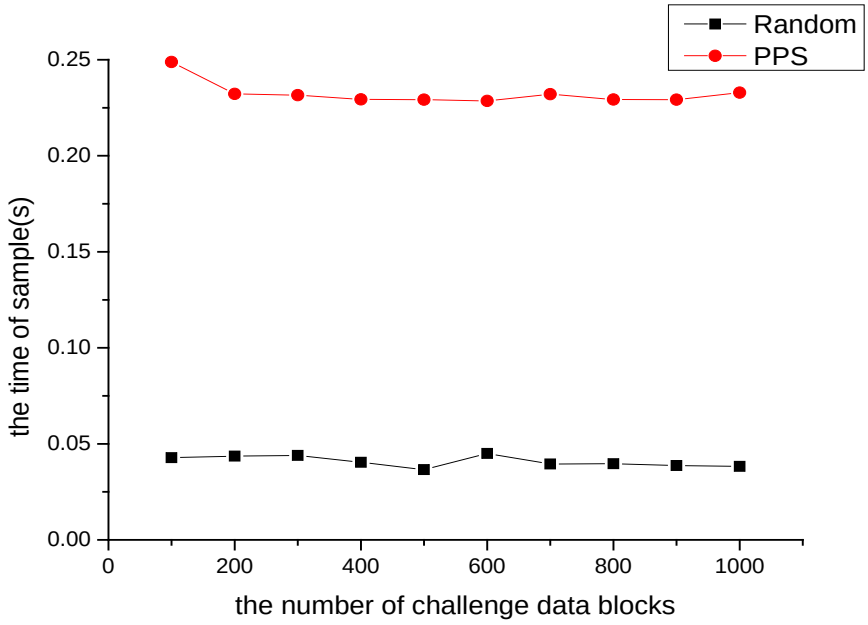

Fig. 4. Computation cost of sample

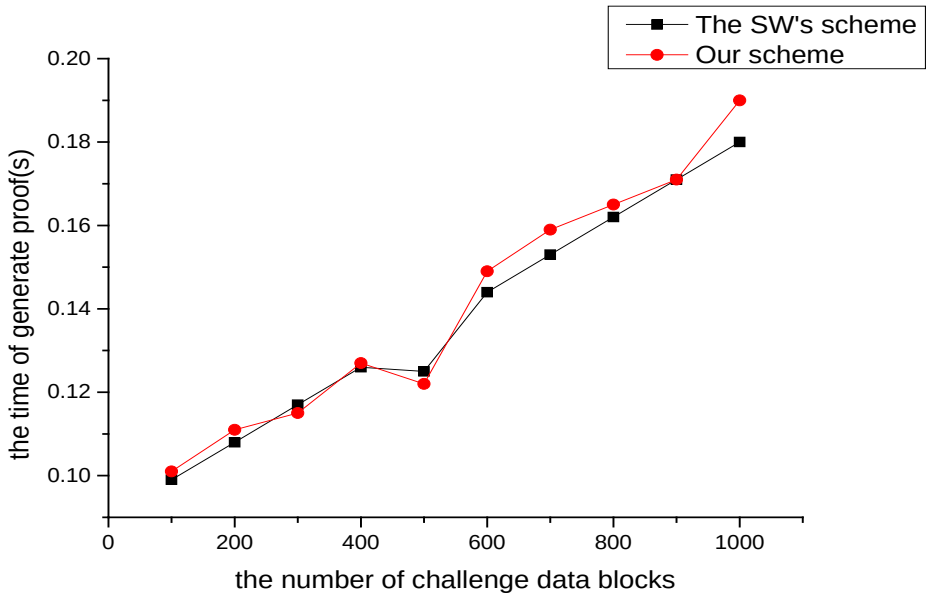

Fig. 5. Computation cost of proof generation

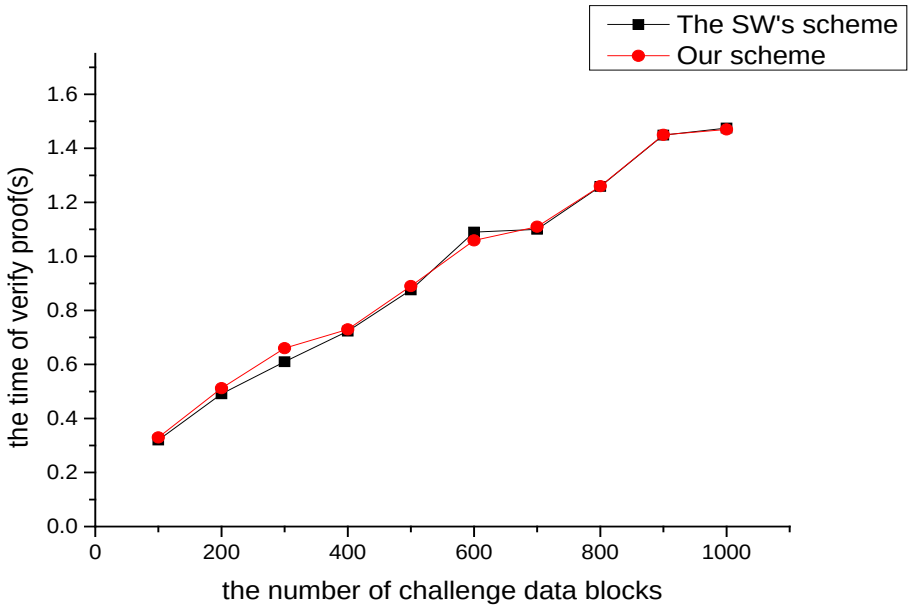

Fig. 6. Computation cost of audit proof

Detection CSP Deleting Data Probability: Assuming that the users store *n* data blocks in the CSP, and the CSP deletes the *l* data blocks. Let *c* is the number of challenge data blocks . Let *X* be a discrete random variable that is defined to be the number of data blocks chosen by *c* that detect the CSP deletes user data behavior.

P_X is the probability that at least one of the data block picked by *c* matches one of the data blocks deleted by the CSP. So:

$$P_X = P\{X \geq 1\} = 1 - P\{X = 0\} = 1 - \frac{n-l}{n} \cdot \frac{n-1-l}{n-1} \cdot \frac{n-2-l}{n-2} \cdot \ldots \cdot \frac{n-c+1-l}{n-c+1} .\text{since}$$

$$\frac{n-j-l}{n-j} \geq \frac{n-j-1-l}{n-j-1}, \text{ it follows that: } 1-(\frac{n-l}{n})^c \leq P_X \leq 1-(\frac{n-j-1-l}{n-j-1})^c$$

P_X is the probability that, if the CSP deletes *l* user data blocks of *n*, then the TPA will detect the CSP malicious behavior with *c* challenge. When the number of *l* is certain, *c* can verify the CSP misbehavior with a certain probability by asking proof for a certain amount of data, independently of the total number of data *n*: e.g., if *l* = 1% of *n*, 460 and 300 data blocks are selected by the TPA in order to achieve P_X of at least 99% and 95%, respectively[7].

Assuming n is 10000 data blocks, the CSP randomly deletes 100 data blocks, which is 1% of n. The PPS and random sampling are used to extract the challenge data blocks c. From **Fig. 7**, it can be seen that the probability of detecting the CSP's malicious behavior by PPS is not significantly different from the probability of detecting by random sampling. When c=100, the probability of using random sampling is 58% and the probability of using PPS is 61%; when c=400, the probability of using random sampling and PPS are 97% and 98 % respectively; when c=500, probability of using random sampling is 100% and the probability of using PPS is also 100%. We can conclude that there is no significantly difference between the probability of PPS and the probability of random in the case of CSP randomly deleting user data.

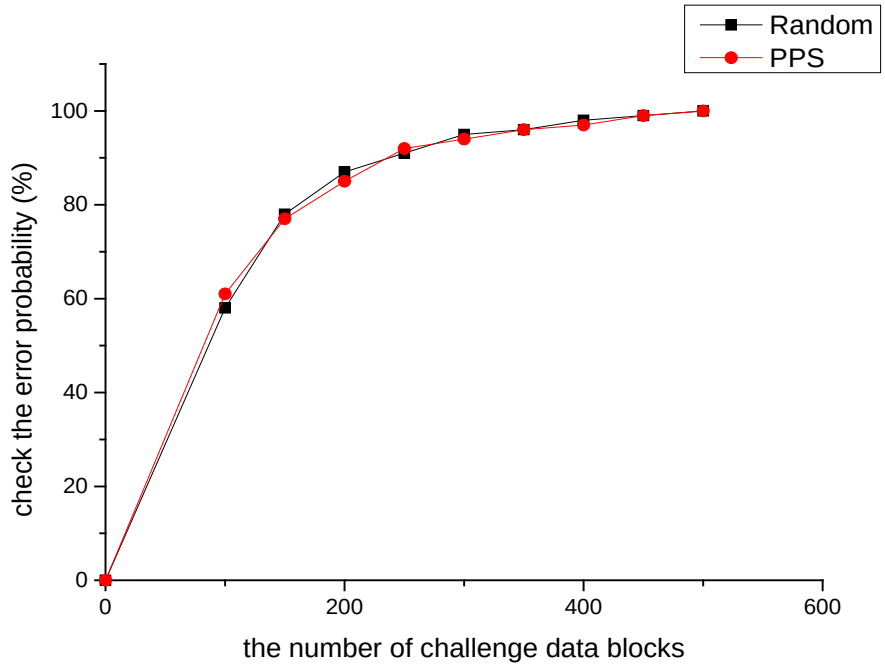

Fig. 7. Comparison of the error probability between PPS and random under the condition of CPS deleting data randomly

Assuming n is 10000 data blocks, the CSP randomly deletes 100 infrequency data blocks, which is 1% of n. The PPS sampling and random sampling are used to extract the challenge data blocks c. **Fig. 8** shows the probability of detecting the CSP's malicious behavior using the PPS and random sampling. It can be seen that the probability of using random sampling is 57% and using the PPS is 78% in the case of c=100. When c=300, probability of using random sampling is 94%, while probability of using PPS is 99%. When c=460, the probability of using random sampling can reach 99%, however, when c=350, the probability of using the PPS is 100%; Obviously, if the CSP randomly deletes infrequency data, the

probability of detecting the CSP's malicious behavior by the PPS is higher than that by random sampling. Specifically, The proposed scheme can use 300 challenge data blocks instead of 460. According to the **Fig. 4**, **Fig. 5** and **Fig. 6**, the computation cost of PPS, proof generation and proof verification are 0.231 seconds ,0.115 seconds and 0.679 seconds in the proposed scheme when c=300, while the computation cost of PPS, proof generation and proof verification are 0.037 seconds ,0.125 seconds and 0.91 seconds in the SW's scheme when c=460. It can be learned the total computation cost of the proposed scheme is smaller than the SW's scheme.

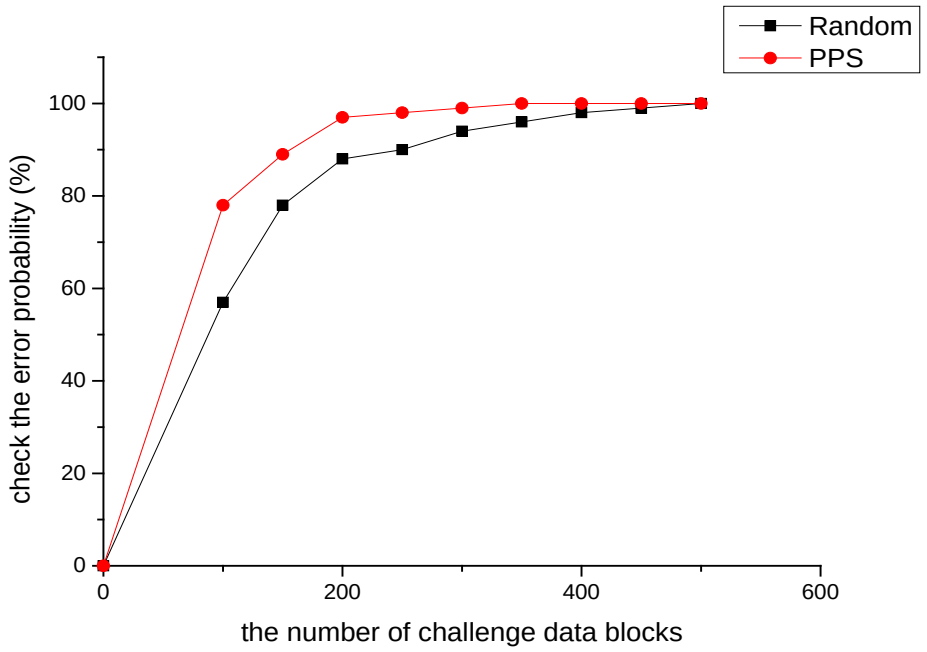

Fig. 8. Comparison of the error probability between PPS and random under the condition of CPS deleting infrequency data

The TPA Collects Auditing Results: The TPA collects auditing results from many users and publishes the aggregated auditing results periodically. Assuming the CSP will delete 85% of user's infrequently used data, the proportion of deleting every user's data is 0.1% -1% of user total data, and then the credit of the CSP is 15%. In this experiment, 200 users are selected, the CSP deletes 170 users' data, and the number of challenge data blocks are 100-1000. The TPA collects auditing results of these 200 users. **Table 2** is the auditing results collected by the TPA. With the increase of challenge data blocks, the probability of checking the CSP delete data gradually increased. When the number of challenge data blocks are 1000, the probability is 83.5%, which is close to the hypothetical CSP deletion probability 85%. The CPS credit is related to the error probability, their sum is 100%. When the deleted data is

between 0.1% to 1%, the probability of checking error is given by **Table 3**. Under the condition of 300 challenge data blocks, the accuracy of checking gradually improved as the number of deleted data increases. This is the reason that the error rate is 69.5%, not 99% with the 300 challenge data blocks.

Table 2. The error probability and credit of the CSP with different challenge blocks

challenge blocks	100	200	300	400	500	600	700	800	900	1000
the error probability	50.5 %	64.5 %	69.5 %	72.5 %	76%	77.5 %	79.5 %	82.5 %	83%	83.5 %
credit of the CSP	49.5 %	35.5 %	30.5 %	27.5 %	24%	22.5 %	20.5 %	17.5 %	17%	16.5 %

Table 3. The check error probability with different ratio of deleting data

the ratio of delete data (C=300)	0.10%	0.20%	0.30%	0.40%	0.50%	0.60%	0.70%	0.80%	0.90%	1%
the check error probability	45.7%	60%	79.25 %	86.25 %	89%	90.75 %	96.25 %	97%	98.75 %	99%

7. Conclusion

In this paper, we propose an improving audit efficiency scheme for cloud storage. The PPS is used to extract the challenge data blocks to handle the situation that the CSP deletes the infrequency data of users. The proposed scheme uses fewer challenge data blocks than the ordinary auditing scheme to achieve the same auditing result. Considering the CSP attaches great importance to his reputation, the TPA collects the auditing results of users and publishes the statistical results regularly. This behavior constrains the server's malicious behavior and extends the auditing cycle, which can reduce the computing pressure of the whole system. The experimental results demonstrate the high efficiency of the proposed scheme.

Acknowledgement

The authors thank the editors and the anonymous reviewers for their valuable comments. This research was supported by National Natural Science Foundation of China (Grant No.61472074,U1708262) and the Fundamental Research Funds for the Central Universities (No.N172304023).

References

[1] Mell P, Grance T, "The NIST definition of cloud computing[J]," *Communications of the Acm*, vol. 5(6), pp. 50-50, 2011. Article (CrossRef Link).

[2] https://en.wikipedia.org/wiki/Cloud_computing

[3] Fox, Armando, et al., "Above the clouds: A berkeley view of cloud computing," *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS 28*, vol. 13 (2009), 2009. Article (CrossRef Link).

[4] Customer Presentations on Amazon Summit Australia, Sydney, 2012, accessed on: March 25, 2013.

[5] D. Zissis and D. Lekkas, "Addressing Cloud Computing Security Issues," *Future Gen. Comput. Syst.*, vol. 28, no. 3, pp. 583-592, Mar. 2011. Article (CrossRef Link).

[6] M. A. Shah, R. Swaminathan, and M. Baker, "Privacy-preserving audit and extraction of digital contents," *Cryptology ePrint Archive*, Report 2008/186, 2008.

[7] G. Ateniese, R. Burns, R. Curtmola, J. Herring,L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proc. of the 14th ACM Conference on Computer and Communications Security*, pp. 598-609, Virginia, USA, 2007. Article (CrossRef Link).

[8] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in *Proc. of the 4th International Conference on Security and Privacy in Communication Netowrks*, pp. 9:1-9:10, Istanbul, Turkey, 2008. Article (CrossRef Link).

[9] C. Wang, Q. Wang, K. Ren, and W. Lou, "Ensuring data storage security in cloud computing," in *Proc. of the 17th International Workshop on Quality of Service (IWQoS'09)*, pp. 1-9, South Carolina, USA, 2009. Article (CrossRef Link).

[10] C. Erway, A. K, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *Proc. of the 16th ACM Conference on Computer and Communications Security*, pp. 213-222, Illinois, USA, 2009. Article (CrossRef Link).

[11] A. Juels and J. Burton S. Kaliski, "Pors: Proofs of retrievability for large files," in *Proc. of the 14th ACM Conference on Computer and Communications Security*, pp.584-597, Virginia, USA, 2007. Article (CrossRef Link).

[12] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Proc. of the 14th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT'08)*, pp 90-107, Melbourne, Australia, 2008. Article (CrossRef Link).

[13] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 5,pp. 847-859, 2011. Article (CrossRef Link).

[14] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for secure cloud storage," *IEEE Transactions on Computers*, vol. 62, no. 2, pp. 362-375, 2013. Article (CrossRef Link).

[15] Wang J, Chen X, Huang X, et al., "Verifiable Auditing for Outsourced Database in Cloud Computing[J]," *IEEE Transactions on Computers*, vol. 64, no. 11, pp. 3293-3303, 2015. Article (CrossRef Link).

[16] C. Liu, C. Yang, X. Zhang, and J. Chen, "External integrity verification for outsourced big data in cloud and iot: A big picture," *Future Generation Computer Systems*, vol. 49, no. 6, pp. 58–67, 2015. Article (CrossRef Link).

[17] J. Yu, R. Hao, H. Zhao, M. Shu, and J. Fan, "IRIBE: Intrusion-Resilient Identity-Based Encryption," *Information Sciences*, vol. 329, pp. 90-104, 2016. Article (CrossRef Link).

[18] Y. Zhang and M. Blanton, "Efficient Dynamic Provable Possession of Remote Data via Balanced Update Trees," in *Proc. of Department of the 8th ACM SIGSAC symposium on Information, computer and communications security. ACM*, pp.183-194, 2013. Article (CrossRef Link).

[19] W.Shen, J.Yu,G. Yang, Y.Zhang, Z.Fu, and R.Hao. "Access-Authorizing and Privacy-Preserving Auditing with Group Dynamic for Shared Cloud Data," *KSII Transactions on internet and information systems*, vol. 10, no. 7, pp. 3319-3338, Jul. 2016. Article (CrossRef Link).

[20] C. Liu, J. Chen, L. T. Yang, X. Zhang, C. Yang,R. Ranjan, and K. Ramamohanarao, "Authorized public auditing of dynamic big data storage on cloud with efficient verifiable fine-grained updates," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 9, pp. 2234-2244, 2014. Article (CrossRef Link).

[21] W. Song, B. Wang, Q. Wang, Z. Peng, and W. Lou, "Tell me the truth: Practically public authentication for outsourced databases with multiuser modification," *Inf. Sci.*, vol. 387, pp. 221–237, May 2017. Article (CrossRef Link).

[22] A.Fu,S.Yu, Y. Zhang, H. Wang, and C. Huang, "NPP: A New Privacy-Aware Public Auditing Scheme for Cloud Data Sharing with Group Users," *IEEE Transactions on Big Data*, pp. (99)1-1, 2017. Article (CrossRef Link).

[23] J.Yu, K. Ren, C. Wang, "Enabling Cloud Storage Auditing With Key-Exposure Resistance," *IEEE transactions on information forensics and security*, vol. 11, no. 6, pp. 1362-1375, june 2016. Article (CrossRef Link).

[24] H.Wang, D. He, J.Yu, Z.Wang, "Incentive and Unconditionally Anonymous Identity-Based Public Provable Data Possession," *IEEE Transactions on Services Computing*, pp. (99)1-1, 2016. Article (CrossRef Link).

[25] Cochran W G, "Samplimg techniques:Willaim Gemmell Cochran.[J]," *New York, John Wiley & Sons*, 1963. Article (CrossRef Link).

[26] A.J.R. Cotter, G. Course, S.T. Buckland, C. Garrod, "A PPS sample survey of English fishing vessels to estimate discarding and retention of North Sea cod,haddock, and whiting," *Fisheries Research*, vol. 55, no. 1-3, pp. 25-35, 2002. Article (CrossRef Link).

[27] Myint T, Htoon M T, Shwe T, "Estimation of leprosy prevalence in Bago and Kawa townships using two-stage probability proportionate to size sampling technique.[J]," *International Journal of Epidemiology*, vol. 21, no. 41, pp. 778-783, 1992. Article (CrossRef Link).

[28] Hoogduin L A, Manager S, Statistician, et al., "Modified Sieve Sampling: A Method for Single-and Multi-Stage Probability-Proportional-to-Size Sampling[J]," *Auditing A Journal of Practice & Theory*, vol. 29, no. 1, pp. 125-148, 2010. Article (CrossRef Link).

[29] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in *Proc. of ESORICS'09, Saint Malo, France*, pp. 355-370, Sep. 2009. Article (CrossRef Link).

[30] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing," in *Proc. of IEEE INFOCOM*, pp. 1-9, 2010. Article (CrossRef Link).

[31] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *Proc. of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT'01)*, pp. 514-532, Gold Coast, Australia, 2001. Article (CrossRef Link).

[32] Midzuno H, "On the sampling system with probability proportionate to sum of sizes[J]," *Annals of the Institute of Statistical Mathematics*, vol. 3, no. 1, pp. 99-107, 1951.

[33] WF Hsien , CC Yang , MS Hwang, "A Survey of Public Auditing for Secure Data Storage in Cloud Computing," *International Journal of Network Security*, vol.18, no.1, pp.133-142, Jan. 2016. Article (CrossRef Link).

[34] Free Software Foundation, The GNU multiple precision arithmetic library, 2015. Article (CrossRef Link).

[35] B. Lynn, The pairing-based cryptographic library, 2015. Article (CrossRef Link).

Kuan Fan is a Ph.D. candidate in the School of computer and Communication Engineering, Northeastern University, Shenyang, China. She received the M.S. degree degree in Nanjing University Of Posts And Telecommunications, Nanjing, China, in 2013. Hers research interests include cloud computing security and big data security.

Mingxi Liu is a M.S. candidate in the School of computer and Communication Engineering, Northeastern University, Shenyang, China. His research interests include cloud computing security and big data security.

Wenbo Shi received the M.S. degree from the Inha University, Incheon, South Korea, in 2007 and the Ph.D. degree from the Inha University, Incheon, South Korea, in 2010. Currently he is a professor at Northeastern University. His research interests include cryptography, network security.