JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 33, 773-787 (2017)
DOI: 10.6688/JISE.2017.33.3.11

Public Auditing Scheme for Data Storage Security in Cloud Computing

T. SUBHA[1] AND S. JAYASHRI[2]
[1]*Department of Information Technology*
Sri Sai Ram Engineering College
Research Scholar, Anna University
Chennai, 600044 India
[2]*Department of Electronics and Communication*
Adhiparasakthi Engineering College
Tamilnadu, 603319 India
E-mail: {subharajan; jayaravi2010}@gmail.com

Cloud computing has become popular and adopted by large number of industries (or) enterprises who outsource their IT services to cloud service provider (CSP). The maintenance and operations of their services have become easier and cost effective. Data processing services are one among the major and important component in IT services, since IT functions progress around data processing. In particular privacy and security of user's data brings many challenges that have not been effectively addressed. A specific problem occurs when clients outsource their files (contains data) to the cloud storage server and the services are offered by cloud service provider (CSP), which is an untrusted entity. In this paper we introduce an Index Table Structure (ITS) based public auditing scheme to verify the integrity of the data stored in a cloud. Previous work requires CSP to store and maintain the data property information needed for auditing. In our approach we keep the data auditing information in trusted third party auditor (TTPA), results in reducing communication overhead and computational cost. Our method is a public auditing scheme based on ITS with TTPA, who performs auditing on behalf of the user. We extend our work to support privacy preserving property by signing the proof generated by cloud server during verification process. Thus, the result shows our proposed scheme is secure and provide with better efficiency.

Keywords: queuing model, scheduling, prioritization, fuzzy decision tree, performance

1. INTRODUCTION

Cloud storage service is receiving more attention nowadays as it become a faster profit growth point owing to its low cost, scalability and a position independent platform for managing client's data [17]. A recent survey shows that IT outsourcing has grown by 79% as companies seek to reduce cost and concentrate on building their core competencies (or) business. The users are thus relieved from the burden of storage management and the maintenance of data. However the users have to depend on their service providers for the continued availability of the data [15]. The best popular ever storage offered by Amazon S3 has also experienced significant downtime [10]. If such an important service is vulnerable to security attacks, it would incur heavy loss to the client's data [1]. The following are the reasons for security risks,

Received July 17, 2016; revised September 2, 2016; accepted September 21, 2016.
Communicated by Balamurugan Balusamy.

- As cloud infrastructures are very powerful devices, they are easily susceptible to both internal and external threats which occur via virtual machine and via system holes [13, 18]. These threats can affect the integrity of the data.
- The cloud service provider behave unfaithfully toward the client or the cloud users to maintain their reputation in case of disputes like he may discard portion of the file (or) temporarily move the file to a slower storage, may delete files which are rarely accessed [1, 13].

Therefore security audit becomes a necessary service to check the integrity, availability and confidentiality of data stored in cloud. Security audit finds in helping the security breaches, to trace back and analyze the various application activities and so on.

The audit services offered for cloud storage servers should provide a valid proof of the integrity of data to the users while compared with traditional audit Service. Data integrity verification without a local copy of data is not directly supported by the well known cryptographic techniques based on signature scheme and hash function [15]. Downloading the whole data for the sake of verification of data integrity has become highly impractical for the audit service. It incurs more communication cost especially for large files.

The security and performance objective to be addressed for outsourced storage in clouds in order to achieve an efficient audit is as follows.

- Public Auditability: Allowing a third party auditor to check the correctness of cloud data at regular intervals (or) based on demand of the user without downloading the whole data. Auditor should not introduce any additional burden to cloud Services. Dynamic Operations – To Support dynamic data operation such as insertion, deletion and modification. It should also be ensured that the validation protocol security cannot be compromised with any security attack.
- Timely detection: To be capable of detecting data losses (or) errors in outsourced storage and other abnormal behaviors *etc.*
- Storage Correctness: To ensure the integrity is verified by server, the one who is providing space for storing and maintaining data.

2. RELATED WORK

Recently, there have been many works [1-15, 19] proposed dealing with integrity and privacy of outsourced data. Ateniese *et al.* [2] is the first to introduce Provable data possession (PDP) model, which detects correctness of clients' outsourced data. Their scheme utilizes RSA-based homomorphic tags for auditing the data and retrieves only linear combination of blocks to verify. The dynamic data operation is not supported by their scheme and hence it works only for static files. Further Ateniese *et al.* [8] construct a dynamic version of their PDP model named scalable and efficient PDP. The downside of the scheme is that it sets a limit on the no of queries raised and do not consider the data storage which is fully dynamic. The above scheme works on symmetric key cryptography. In [3] Juels *et al.* present a proof of retrievability (POR) scheme that ensures the retrievability of files and possession. They used spot checking by embedding special blocks called sentinels and error correcting codes to ensure the possession of files. This

scheme is also bound to a limited number of queries and Public auditability is also not achieved. Shacham and waters [4] construct a scheme with full security proof based on the scheme defined in [3]. BLS signatures built on homomorphic linear authenticators were introduced in their scheme [21]. These signatures can be aggregated into a single signature and also provides a small authenticated value for public verifiability. But still, authors considered only static files and do not support dynamic data operations. The collected proof responses that are obtained during the auditing process provide a way for the verifier to derive the data. The framework for exploring dynamic provable data construction is introduced by Erway *et al.* [9]. The dynamic version of PDP [2] model is extended in the model [9] helps to support provable updates to files that are stored using rank based skip authentication lists. This scheme is fully dynamic version of PDP scheme. Block insertion is efficiently supported with authenticated skip list data structure by eliminating index tag computation in Ateniese [2] model. Wang *et al.* [1, 7, 15] proposed a dynamic data storage in a distributed scenario. They utilized challenge – response scheme in order to locate the possible errors and to find the correctness of data. However their scheme supports partial dynamic data operations.

The existing system [1-16] focuses on checking the integrity verification in cloud storage systems. We list and give a comparison of existing techniques, involving provable data possession, scalable PDP, Dynamic PDP, Compact proof of retrievability. But still the problem of ensuring both public auditability and supporting full dynamic data operations has not been fully addressed. Hence designing an efficient and secure data storage system has become an open challenge task in cloud storage. We list and give a comparison of existing techniques [1-10], involving provable data possession, scalable PDP, Dynamic PDP, Compact proof of retrievability in Table 1.

Table 1. Different schemes comparison for a file with *n* blocks.

Scheme	CSP computation	Client computation	Communication cost	Fragmentation	Privacy	Dynamic operations			Prob. of Detection
						Modify	Insert	Delete	
PDP [2]	$O(t)$	$O(t)$	$O(1)$		✓				$1-(1-\rho)^t$
SPDP [8]	$O(t)$	$O(t)$	$O(t)$	✓	✓	✓		✓	$1-(1-\rho)^t$
DPDP-I [9]	$O(t \log n)$	$O(t \log n)$	$O(t \log n)$		✓	✓	✓	✓	$1-(1-\rho)^t$
CPOR-I [4]	$O(t)$	$O(t)$	$O(1)$						$1-(1-\rho)^t$
CPOR-II [4]	$O(t+s)$	$O(t+s)$	$O(s)$	✓					$1-(1-\rho)^t$
DPDP-II [9]	$O(t \log n)$	$O(t \log n)$	$O(t \log n)$			✓	✓	✓	$1-(1-\rho)^{\Omega(n)}$
Dynamic audit	$O(t+s)$	$O(t+s)$	$O(s)$	✓	✓	✓	✓	✓	$1-(1-\rho)^{t.s}$
Our scheme	$O(t)$	$O(t)$	$O(t)$		✓	✓	✓	✓	$1-(1-\rho)^t$

't' – no of sampled blocks, 's'– sectors in each block, 'ρ' – probability of block corruption

Thus our contribution can be summarized as follows:

Design Goals:

- To perform public auditing with the help of trusted third party auditor (TTPA) to ensure the correctness of data.
- To secure the data from active adversaries by applying signature to the data before it is being sent from server to auditor during the verification process.

3. DESIGN OF OUR SCHEME

3.1 Preliminaries

Bilinear Map

Let e: $G \times G \rightarrow GT$. It is a bilinear map, where G and GT are groups of prime order p. Let g is the generator of G. H: $\{0, 1\} \rightarrow G$ is a cryptographic hash function, which maps strings to G, can be viewed as a random oracle [16]. A map e: $G_1 \times G_2 \rightarrow G_2$ is called a bilinear pairing if it satisfies the following three conditions [21].

1. It is computable: there should be an algorithm to compute e efficiently;
2. Bilinear: for all $h_1, h_2 \in G$, and $a, b \in Z_p$, $e(h_1^a, h_2^b) = e(h_1, h_2)^{ab}$;
3. Non-degenerate: $e(g, g) \neq 1$, g is a generator of G.

3.2 System Model

Our proposed system model consists of three main entities named Clients, Cloud Storage Servers (CSS) and Trusted Third Party Auditor (TTPA). The architecture of our system is shown in Fig. 1.

- Clients: users have large amounts of data to outsource onto a cloud. They can leverage the maintenance operations on the data to the cloud.
- Cloud Storage Server (CSS): Cloud service providers have an enormous amount of storage space to store client's data.
- Trusted Third Party Auditor (TTPA): Anyone who has expertise and capabilities to check the integrity of the data stored in cloud based upon the request from the user. The audit log is sent to the user. We assume that the TTPA is trusted and server is un-trusted and semi trusted.

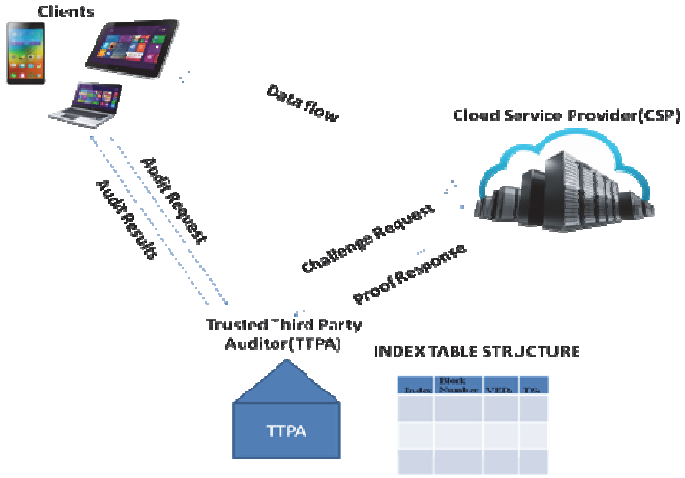

Fig. 1. System architecture for data integrity audit.

3.2.1 Algorithm

Merkle Hash Tree and Index Hash Table are the different authentication structures was introduced in earlier works carried out by many researchers. MHT (Merkle Hash Tree Structure) shown in Fig. 2 was introduced by wang *et al.* [1, 13] and Index Hash Table was introduced by Yan Zhu *et al.* [10] is given in Table 2. When hash trees are used for authentication checking, generating proof becomes complex and time consuming. Moreover the communication overhead and the incurred computation cost are high in those systems. The scheme by Zhu *et al.* [10] uses a data structure called index hash table (IHT) in hybrid cloud is shown in Table 2. It generates hash values and records any changes in data blocks during the verification process. Their scheme also helps to reduce computational cost and communication overhead by storing values in TPA instead of storing in cloud service provider (CSP). But the updating operation requires modification of averaging $N/2$ elements, where N denotes the total number of blocks [10]. This is because of the sequence structures used in index hash table. Also their scheme additionally requires regeneration of block tags since the block numbers of some blocks are modified inevitably. It creates unnecessary overhead in communication cost, and causes additional computation cost. Hence it is not efficient. We design a new table structure named ITS (Index Table Structure), in order to improve an efficiency of the auditing mechanism.

Table 2. Index hash table.

No.	B_i	V_i	R_i	
0	0	0	0	used to head
1	1	2	r'_1	Update
2	2	1	r_2	
3	4	1	r_3	Delete
4	5	1	r_5	
5	5	2	r'_5	Insert
:	:	:	:	
N	N	1	r_n	
n+1	n+1	1	R_{n+1}	Append

No. – serial number, R – random integer,
B_i – block no, V_i – version

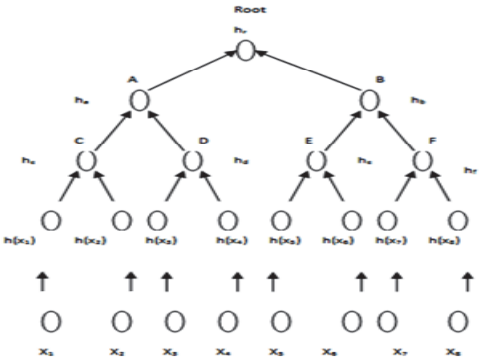

Fig. 2. Merkle hash tree.

In this paper, we consider a new table structure Index Table Structure (ITS) for public auditing is represented in Table 3. Our proposed table contains the following elements as index no; block Number, version, timestamp and its memory reference. In our system, TTPA maintains index table structure (ITS) as specified in Yan Zhu *et al.* [10]. Their scheme checks the integrity of data in a hybrid cloud. We assume the TTPA is trusted and he/she is not able to read the data contents of users. This helps us to preserve privacy protection. In our proposed system, the user sends index information to Index Table Structure (ITS) maintained by Trusted Third Party Auditor (TTPA). The index information contains version number (VER_i) and timestamp (TS_i) of a particular block. Then signature σ_i is generated during the setup phase. The generated signature consists of both block number (m_i) and version number (VER_i) and timestamp (TS_i). TTPA verifies the version number and the timestamp corresponding to challenged blocks in verifi-

cation phase. This step is performed to check whether the proof is correct or not (Eq. 8). Our proposed system aims to reduce the communication cost and computation overhead.

Table 3. Index table structure (ITS).

Index no	Block Number (b_i)	VER_i	TS_i
1	1	VER_1	TS_1
2	2	VER_2	TS_2
3	3	VER_3	TS_3
⋮	⋮	⋮	⋮
n–1	n–1	VER_{n-1}	TS_{n-1}
N	N	VER_n	TS_n

3.3 Proposed System

Initially file F to be outsourced is preprocessed and is divided in to n blocks.

$$F = \{b_1, b_2, \ldots, b_n\} \text{ where } \{b_i\}_{1 \le i \le n} \tag{1}$$

We can further divide each block into s sectors

$$b_i = \{b_{i,1}, b_{i,2}, \ldots, b_{i,s}\} \tag{2}$$

where $\sum_{j=1}^{s} m_{i,j}$, in order to minimize the cost required for extra storage and space.

Our scheme is categorized into two phases based on Index Table Structure (ITS). They are setup phase and proof verification phase. The setup phase is explained as below.

Key Generation

The client generates a key pair (S_{sk}, S_{pk}) for signing. Then it chooses a random value $y \leftarrow Z_p$. From this, user computes $a \leftarrow g^y \in G$. The secret key is $SK = \{y, S_{sk}\}$ and the public keys are chosen as $PK = \{S_{pk}, a, g, u\}$ where g and u are random elements.

Signature Generation

First we compute file tag for the given file $F = b_i$ ($i = 1, 2, \ldots, n$).

The file tag can be used to identify a particular file during integrity verification. The tag can be calculated as

$$\text{T} = \text{filename}||n||\text{sig}_{Ssk}(\text{filename}||n) \tag{3}$$

where n is number of blocks. $\text{sig}_{Ssk}(\text{filename}||n)$ denotes the file name and no of blocks are signed using secret key. In addition to this, the user sends the following information (T, VER_i, TS_i) to the TTPA during setup phase. Then client computes signature (σ_i) for each block $m_i \in Z_p$ with the help of public key u.

$$\sigma_i = (H(m_i||VER_i||TS_i).u^{(m_i||VER_i||TS_i)})^y \tag{4}$$

Let ϕ denote the set of signatures for all the blocks as $\phi = \{\sigma_i\}$, $1 \le i \le n$. Then client sends the following information $\{F, \phi = \{\sigma_i\}_{1 \le i \le n}, \mathbf{T}\}$ to cloud service provider (CSP). User deletes the file and corresponding signatures from the local storage.

Algorithm: Setup phase

```
1 begin
2     data owner chooses a key pair (S_sk, S_pk) for signing
3     choose a random value y ← Z_p; compute a ← g^y ∈ G.
4     secret key SK = {y, S_sk}; public key PK = {S_pk, a, g, u}
5     compute file tag T = filename||n||sig_Ssk (filename||n)
6     send (T, VER_i, TS_i) to TTPA. TTPA adds it to ITS.
7     for round blocks (i ← 1 to n) do
8         begin
9             compute signature tag σ_i = (H(m_i||VER_i||TS_i).u^(m_i||VER_i||TS_i))^y
10        end
11    denote set of signatures φ = {σ_i}, 1 ≤ i ≤ n
12    send {F, φ = {σ_i}_1≤i≤n, T} to CSP
13 end
```

Proof verification phase is explained as below:

- **Audit Process**

The TTPA can verify the integrity of the data by sending challenge request to the server. Initially TTPA verifies the file tag T of a file F using public key S_{pk}. It checks the validity of file and TTPA quits if fail. Otherwise it recovers u. If the proof on validity is correct, TTPA constructs a challenge message and sends the challenge request to the server. It proceeds to next step as shown below.

- **Generation of Challenge Request**

To generate message challenge TTPA picks INDEX = {INDEX$_i$} $1 \le i \le c$, random c number subset $S = \{s_l \mid l \in \text{INDEX}\}$. INDEX = {INDEX$_i$} $1 \le i \le c$ specifies the all blocks index set and $S = \{s_l \mid l \in \text{INDEX}\}$ specifies total no of blocks to be verified. This subset indicates the block positions to be checked. TTPA sends the request to CSP.

Server responds with the proof upon receiving the challenge message,

$$\text{Block Proof } \mathbb{P} = \sum_{i=s_1}^{s_c} X_i m_i \in Z_p. \tag{5}$$

It represents linear combination of all the sampled blocks.

$$\text{Authenticator proof } \Psi = \prod_{i=s_1}^{s_c} \sigma_i^{x_i} \in G \tag{6}$$

It represents all tags aggregated authenticator values.

Both proofs are aggregated in to single value and CSP chooses a random value $r \leftarrow Z_p$. It calculate $R = e(u, a)^r$. A blind factor $\mu' = r + \gamma\mu$ is also calculated. CSP responds with the following proof,

$$Proof = \{\mu, \Psi, R\}. \tag{7}$$

There is a possibility for any adversary to modify the data by observing the conversations between CSP and TPA as specified in [11, 12]. CSP digitally signs (§) the generated proof using secret key by applying elliptic curve digital signature algorithm [20] before sending it to the auditor to overcome this problem. Finally this is sent to the TTPA.

$$Proof = \{\mu, \Psi, R, \S\} \tag{8}$$

Correctness of Proof Verification

TTPA verifies the validity of proof as follows, The TTPA verifies the sign using public key. If it is valid, then it computes challenged blocks hash values $H = \Pi H(\mathrm{VER}_i, \mathrm{TS}_i)^{Xi}$ and further TTPA verifies the validity of proof by checking the equation given below.

$$R.e(\sigma^{\gamma}, g) = e((\prod_{i=s_1}^{s_c} H(m_i \,||\, VER_i \,||\, TS_i)^{x_i})^{\gamma} .u^{\mu}, a) \tag{9}$$

It outputs TRUE if the equation holds, otherwise emits FALSE. The correctness of the verification equation can be verified as follows.

$$R.e(\sigma^{\gamma}, g) = e(u, a)^r .e((\prod_{i=s_1}^{s_c} H(m_i \,||\, VER_i \,||\, TS_i).u^{(m_i||VER_i||TS_i)x_i y})^{\gamma}, g)$$

$$= e(u^r, a).e((\prod_{i=s_1}^{S_c} H(m_i \,||\, VER_i \,||\, TS_i)^{x_i} .u^{(m_i||VER_i||TS_i)^{\gamma x_i}}), g)^y$$

$$= e(u^r, a).e((\prod_{i=s_1}^{S_c} H(m_i \,||\, VER_i \,||\, TS_i)^{x_i})^{\gamma} .u$$

$$= e((\prod_{i=s_1}^{s_c} H(m_i \,||\, VER_i \,||\, TS_i)^{x_i})^{\gamma} .u^{\gamma+r}, a)$$

$$= e((\prod_{i=s_1}^{s_c} H(m_i \,||\, VER_i \,||\, TS_i)^{x_i})^{\gamma} .u, a)$$

Algorithm: Verification phase

1 begin
2 **TTPA generates challenge request**
3 begin
4 TTPA constructs a challenge message as chal = (INDEX={INDEX$_i$}$1 \le i \le c$, $S = \{s_l | l \in \mathrm{INDEX}\}$).

Send challenge message "challenge" to CSP
end
CSP computes proof
begin
$\mu = \sum_{i=s_1}^{S_c} X_i m_i \quad \Psi = \prod_{i=s_1}^{s_c} \sigma_i^{x_i}$
choose a random value $r \leftarrow Z_p$.
calculates $R = e(u, a)^r$. apply blind factor $\mu' = r + \gamma\mu$
generate Proof $\S = \{\mu, \Psi, R\}$
sign the proof and send $\{\mu, \Psi, R, \S\}$ to TTPA
end
TTPA verifies the proof
Checks the signature on proof $\{\mu, \Psi, R, \S\}$
If (Result=PASS)
begin
checks hash value $H=\Pi H(\mathrm{VER}_i, \mathrm{TS}_i)^{X_i}$
verifies signature by checking Eq. (8)
if (output==TRUE) emits PASS
else emits FALSE
end
else QUIT
end

4. DYNAMIC OPERATIONS

Our system supports dynamic operations such as insertion of data (DI), modification of data (DM) and deletion of data (DD).

Data Insertion (DI) – This protocol supports insertion of a new block b_{new} after a given block b_i into a file F.

Algorithm: Data Insertion (DI)

begin
Update record in ITS
begin
user generates new data information (VER_{new}, TS_{new}) for the new block b_{new}.
sends update request $\mathrm{update}_{ins}(F, \mathrm{DI}, i, \mathrm{VER}_{new}, \mathrm{TS}_{new}) \rightarrow$ TTPA
end
TTPA performs the following after receiving update_{ins} request
begin
TTPA finds the last record in ITS and inserts the new one after it
update both VER_{new}, TS_{new}, increment pointer by 1
end
update stored data in cloud

```
13      begin
14      user generates new signature σ_new for the new block b_new based on Eq. (4)
15      sends update request update_ins(F, DI, i, b_new (VER_new, TS_new), σ_new → CSP
16      CSP generates new version of file F_new and tag set T_new = F_new||n||sig_Ssk(F_new||n)
17      end
18 end
```

Data Deletion (DD) – This protocol supports deletion of a particular block (b_i) and requires moving all the blocks after performs deletion.

Algorithm: Data Deletion (DD)

```
1  begin
2     update record in ITS
3     begin
4        user sends update request update_del(F, DD, i) → TTPA
5        TTPA deletes ith record in ITS
6        decrement pointer by 1
7     end
8     update stored data in cloud
9     begin
10    user sends update_del(F, DD, i) → CSP
11    CSP gets new file version F_new and tag set T_new = F_new||n||sig_Ssk(F_new||n)
12    end
13 end
```

Data Modification (DM) – This protocol supports the replacing of a specified block b_i with a new one b'_{new}. This is the most frequently used operations in cloud. It replaces the specified block of b_i to b'_{new}.

Algorithm: Data Modification

```
1  begin
2     user generates new (VER'_new, TS'_new) information for the block b'_new
3     update record in ITS
4        begin
5        Sends update request update_mod(F, DM, i, VER'_new, TS'_new) for b'_new → TTPA
6        end
7     TTPA performs the following after receiving update_mod request
8        begin
9              TTPA finds the appropriate record in ITS
10             replace (VER, TS) with VER'_new, TS'_new
11       end
12    update stored data in cloud
13       begin
14       user generates new signature σ'_new for the new block b'_new based on Eq. (4)
```

15 sends update modification request $update_{mod}(F, DM, i, b_{new}(VER_{new}, TS_{new}), b_{new}(VER'_{new}, TS'_{new}), \sigma'_{new} \rightarrow CSP$

16 CSP generates new version of file F'_{new} and tag set $T'_{new} = F'_{new} ||n||sig_{ssk}(F'_{new}||n)$ after replacing the old block with a new one

17 end

18 end

Table 4. Illustration of dynamic operations.

Insert(b_i, b_{new})

Index no	Block Number (b_i)	VER_i	TS_i
1	1	VER_1	TS_1
2	2	VER_2	TS_2
3	3	VER_3	TS_3
⋮	⋮	⋮	⋮
b_i	b_i	VER_{bi}	TS_{bi}
⋮	⋮	⋮	⋮
n–1	n–1	VER_{n-1}	TS_{n-1}
n	n	VER_n	TS_n
b_{new}	b_{new}	VER_{bnew}	TS_{bnew}

Insert(b_i, b_{new})-after insertion

Index no	Block Number (b_i)	VER_i	TS_i
1	1	VER_1	TS_1
2	2	VER_2	TS_2
3	3	VER_3	TS_3
4	4	VER_4	TS_4
⋮	⋮	⋮	⋮
b_i	b_i	VER_i	TS_i
b_{new}	b_{new}	VER_{bnew}	TS_{bnew}
b_i+1	b_i+1	VER_{bi+1}	TS_{bi+1}
⋮	⋮	⋮	⋮
n–1	n–1	VER_{n-1}	TS_{n-1}
n	n	VER_n	TS_n
n+1	n+1	VER_{n+1}	TS_{n+1}

Delete(b_i)-after deletion

Index no	Block Number (b_i)	VER_i	TS_i
1	1	VER_1	TS_1
2	2	VER_2	TS_2
3	3	VER_3	TS_3
4	4	VER_4	TS_4
⋮	⋮	⋮	⋮
b_i–1	b_i–1	VER_{bi-1}	TS_{bi-1}
b_i	b_i	VER_{bi}	TS_{bi}
⋮	⋮	⋮	⋮
n–1	n–1	VER_{n-1}	TS_{n-1}
n	n	VER_n	TS_n

5. SECURITY AND PERFORMANCE ANALYSIS

This section deals about security and performance analysis of our proposed system. We prove that our scheme is sound and correct.

5.1 Security Analysis

We present security analysis of our proposed system as below.

Theorem: Unforgeability of the proof

In our scheme it is computationally infeasible for the adversary to forge an auditing proof to pass the verification.

***Proof*:** In our system TTPA sends a challenge request chal = (INDEX={$INDEX_i$} $1 \le i \le c$, $S = \{s_l | l \in INDEX\}$) to CSP. CSP responds with the following proof {ℼ, Ψ, R} for the data file F in order to pass the verification. CSP signs the data with secret key (PK) using digital signature algorithm to protect the signature {ℼ, Ψ, R, §} and data from active adversaries. TTPA verifies signature using public key (PK). This serves the authentication purpose to prove the proof is signed by provider. If any attacker tries to alter the data, the same cannot be done since it is signed. This proves our scheme is secured.

5.2 Performance Analysis

We compare our scheme with the previous schemes by Wang [13] using Merkle-Hash Tree, Y. Zhu [10] using IHT.

5.2.1 Experimental results

We have conducted our experiment in a Eucalyptus environment. A private cloud has been setup using a eucalyptus open source infrastructure. This open source platform can be used to test IaaS (Infrastructure as a service) related cloud computing problems [22]. We can able to experience the real cloud setup and benefits by conducting experiments in eucalyptus tool. We have installed eucalyptus fast start version 3.4.1 on Centos6 in an Intel core i5-3520 CPU at 2.2 GHz, 500 GB SATA drive and 8 GB RAM. Our algorithms have been implemented in this setup and the results are discussed here.

5.2.1.1 Storage cost

Normally the file F and the no of blocks denoted as n is taken for analyzing storage cost in CSP and TPA. The storage cost in TPA is nil and CSP includes the cost incurred for storing the tags and the metadata used for auditing. In the above discussed schemes, there is no storage cost in TPA and the cost of CSP is given as $n.\mu$ where n is the no of blocks and μ is the bit length. Our scheme stores tags in CSP and metadata in TTPA. So the cost is $v(n)$.

5.2.1.2 Computation cost

Computation cost involves computation of expensive operations. We present the computation cost incurred for setup phase, verification phase and cost incurred for dynamic operations phase.

Setup phase

Here the time taken to process users request based on block numbers. We derive from the result that the time taken for processing is directly proportional to block numbers. Our scheme consumes less time to execute compared with the earlier systems. It is shown in Fig. 4.

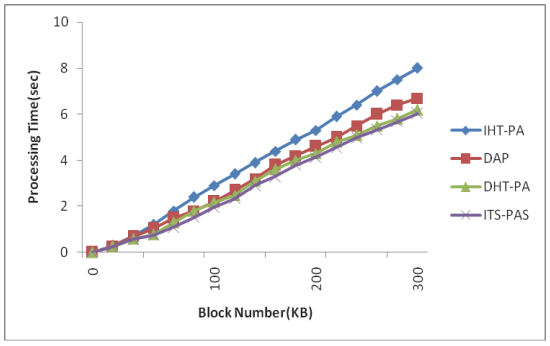

Fig. 4. Setup phase – processing time vs block number (different sizes).

Verification phase

The proposed scheme cost for verification is much closer to the schemes by Wang [13] and Y. Zhu [10]. It is represented in Fig. 5.

Dynamic operations phase

The time taken for performing block updation cost is almost similar to other systems mentioned. The time for operations on blocks will increase as the file size increases. However the time required performing block operation is less. It is shown in Figs. 6 and 7.

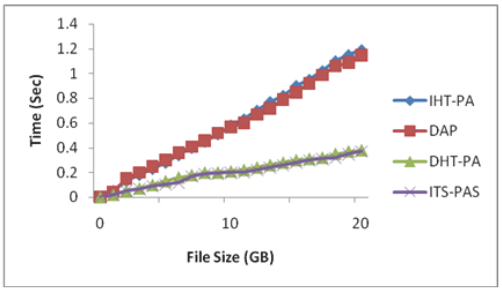

Fig. 5. Verification phase (time vs block size).

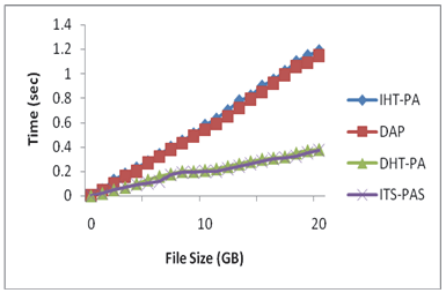

Fig. 6. Block insertion.

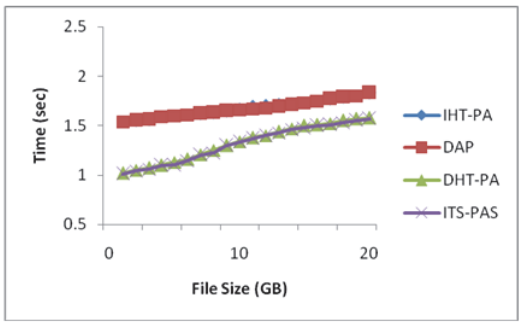

Fig. 7. Block deletion.

6. CONCLUSION

The clients or data owner needs to frequently check their data outsourced on to a remote server. The cloud storage server (CSS) maintained by CSP (cloud service provider) is an untrusted entity. So this auditing task is delegated to trusted third party auditor (TTPA) who is capable of checking the integrity proof of data and confidentiality by running verification algorithm. TTPA performs auditing process at regular intervals to detect the integrity breach of data and the audit report is sent to the user. This type of auditing is called public auditing. In this paper we achieved public auditability and dynamic data operations support. Thus our proposed system works based on Index Table Structure (ITS) to perform auditing. In addition to this we explore digital signature technique to sign the proof generated by a server during the auditing process. The detailed analysis is explained and the efficiency is proved. This is to thwart the attacks involved

by external adversaries. They interacts the message between cloud server and TPA. He modifies the content of the data and fools the server to believe the data is well maintained by the server. Our simple solution to solve this issue is to implant a small piece of software (or) program that alerts the server as well as the user if the data has been accessed by adversaries. We achieved a solution to remedy this weakness by applying elliptic curve digital signature algorithm and a piece of program in the existing system so as to maintain the efficiency, cost and desirable properties of an existing system.

REFERENCES

1. Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in *Proceedings of European Symposium on Research in Computer Security*, 2009, pp. 355-370.
2. G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proceedings of ACM Conference on Computer and Communications Security*, 2007, pp. 598-609.
3. A. Juels and B. S. Kaliski, Jr., "Pors: proofs of retrievability for large files," in *Proceedings of ACM Conference on Computer and Communications Security*, 2007, pp. 584-597.
4. H. Shacham and B. Waters, "Compact proofs of retrievability," in *Proceedings of Annual International Conference on Theory and Applications of Cryptology and Information Security*, 2008, pp. 90-107.
5. K. D. Bowers, A. Juels, and A. Opera, "Proofs of retrievability: Theory and implementation," Cryptology ePrint Archive Report 2008, pp. 90-107.
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. Q. Wang, K. Ren, W. Lou, and Y. Zhang, "Dependable and secure sensor data storage with dynamic integrity assurance," in *Proceedings of IEEE INFOCOM*, 2009, pp. 954-962.
8. G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in *Proceedings of ACM SEcureComm*, 2008, pp. 1-10.
9. C. Erway, A. Kupku, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *Proceedings of ACM Conference on Computer and Communications Security*, 2009, pp. 213-222.
10. Y. Zhu, H. Wang, Z. Hu, G.-J. Ann, H. Hu, and S. S. Yau, "Dynamic audit services for outsourced storage in clouds," *IEEE Transactions on Services Computing*, Vol. 6, 2013, pp. 227-238.
11. B. Wang, B. Li, and H. Li, "Oruta: privacy preserving public auditing for shared data in the cloud," in *Proceedings of IEEE International Conference on Cloud Computing*, 2012, pp. 293-302.
12. B. Wang, B. Li, and H. Li, "Knox: privacy preserving auditing for shared data with large groups in the cloud," in *Proceedings of the 21st Australasian Conference on Information Security and Privacy*, 2012, pp. 507-525.
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 Par-*

allel Distributed Systems, Vol. 22, 2011, pp. 847-859.
14. K. Yang and X. Jia, "An efficient and secure dynamic auditing protocol for data storage in cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 24, 2013, pp. 1717-1726.
15. C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. J. Lou, "Privacy-preserving public auditing for secure cloud storage," http://eprint.iacr.org/2009/579.pdf.
16. M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *Proceedings of ACM Conference on Computer and Communications Security*, 1993, pp. 62-73.
17. National Institute of Standards and Technology, Digital Signature Standard (DSS), Federal information processing standards publication 186-2, US Department of Commerce/NIST, 2000.
18. C. Rong, S. T. Nguyen, and M. G. Jaatun, "Beyond lightning: a survey on security challenges in cloud computing," *Computers and Electrical Engineering*, Vol. 39, 2013, pp. 47-54.
19. R. C. Merkle, "Protocols for public key cryptosystems," in *Proceedings of IEEE Symposium on Security and Privacy*, 1980, pp. 122-134.
20. "Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA)," ANSIX9.62, 1999.
21. D. Boneh, B. Lynn, and H. Scacham, "Short signatures from the weil pairing," in *Proceedings of 7th International Conference on the Theory and Application of Cryptology and Information Security*, 2001, pp. 514-532.
22. https://www.usenix.org/conference/lisa-09/eucalyptus-open-source-infrastructure-Cloud-computing.

T. Subha received her M.Tech degree in Information Technology from Sathyabama University, Chennai, India. She is currently pursuing the Ph.D. degree in the Department of Information and Communication Engineering, Anna University, India. She is currently working as Associate Professor in Information Technology in Sri Sai Ram Engineering College, India. Her research interests include network security and algorithms, data storage security in cloud computing.

S. Jayashri is a full-time Professor in ECE and Director of the Institution in Adhiparasakthi Engineering College, India. She received her B.E. degree from Madurai Kamaraj University, India. She received her M.E and Ph.D. degree from Anna University, India. She is a member in ISTE, IETE, IASTE, and IAE. She has published more than 72 papers in National and International Conference/Journals. Her research interest includes, wireless communication, mobile communication, optical communication, VLSI design, cognitive radio and cloud computing.