Identity-based Provable Data Possession for Multicloud Storage with Parallel Key-Insulation

S. Mary V. Nithya[1*] and Dr.V. Rhymend Uthariaraj[1]
[1] Ramanujan Computing Centre, Anna University, Guindy, Chennai – 600 025, Tamil Nadu, India.
[e-mail : nithya.roshan08@gmail.com ; rhymend@annauniv.edu]
[*]Corresponding author: S. Mary V. Nithya

Received October 27, 2020; revised February 24, 2021; accepted August 22, 2021; published September 30, 2021

Abstract

Cloud Storage is the primary component of many businesses on cloud. Majority of the enterprises today are adopting a multicloud strategy to keep away from vendor lock-in and to optimize cost. Auditing schemes are used to ascertain the integrity of cloud data. Of these schemes, only the Provable Data Possession schemes (PDP) are resilient to key-exposure. These PDP schemes are devised using Public Key Infrastructure (PKI-) based cryptography, Identity-based cryptography, etc. PKI-based systems suffer from certificate-related communication/computational complexities. The Identity-based schemes deal with the exposure of only the auditing secret key (audit key). But with the exposure of both the audit key and the secret key used to update the audit key, the auditing process itself becomes a complete failure. So, an Identity-based PDP scheme with Parallel Key-Insulation is proposed for multiple cloud storage. It reduces the risk of exposure of both the audit key and the secret key used to update the audit key. It preserves the data privacy from the Third Party Auditor, secure against malicious Cloud Service Providers and facilitates batch auditing. The resilience to key-exposure is proved using the CDH assumption. Compared to the existing Identity-based multicloud schemes, it is efficient in integrity verification.

***Keywords*:** auditing, cloud data, integrity, key-exposure, Provable Data Possession

1. Introduction

Cloud storage has become a vital resource. Several enterprises are moving to cloud for business related data storage and applications. The need has massively increased these days due to the COVID-19 pandemic [1]. AWS, Azure, Dropbox, Google Drive, IBM Cloud are some of the prominent Cloud Service Providers (CSPs) who are serving enterprises. To deal with data growth and vendor-lock in, to optimize cost and performance [2, 3], many large enterprises adopted a multicloud strategy which can be multiple public or multiple private or hybrid cloud. Some of them store sensitive data such as business plans, financial records etc. even in public clouds. Despite the benefits of multicloud storage, there are challenges of confidentiality, integrity, availability etc. Of all the challenges, data integrity [4] is the principal focus of cloud clients since loss of data can damage the business of the client.

1.1. Auditing

The integrity of cloud data is affected by many factors, which may be accidental or intentional [5]. Hardware/software failures, human errors and misdeeds can damage the data. So, it is the duty of the cloud clients to audit the data periodically [6] and ensure that it is neither altered nor deleted. Most of the schemes for cloud data auditing use the Provable Data Possession (PDP) model [7]. This model eliminates the download of actual data from the cloud called blockless verification. Blockless verification reduces the communication cost of data download.

To enable auditing, each data block stored in the cloud has an authentication tag signed using the client's secret key called the auditing secret key or the audit key. Exposure of the audit key is a critical issue since it forsakes the purpose of auditing itself. Both the semi-trusted CSP and the malicious CSP may try to acquire the key. Enterprises also run through the security risk of managing cryptographic keys. With the key, a semi-trusted CSP can attempt to hide from the client, the unforeseen failures like data loss inorder to avoid losing status and profits. A malicious CSP may intentionally delete the infrequently accessed data to allure additional clients and still successfully pass during the audit [8]. Key disclosure is mostly undetectable or detected only after some serious impairment. It is also unavoidable. So, it is necessary to minimize the impact of key- exposure even before being informed about the exposure.

The audit key exposure issue is handled in some of the PDP schemes (Public Key Infrastructure (PKI-) based schemes and Identity-based schemes) for single cloud storage. In these schemes, the audit key evolves over time i.e., the lifetime of the data file is partitioned into time periods and the audit key is updated at the beginning of time periods [8-15]. Some of these schemes are secure in the preceding time periods of the exposed key (forward security). The remaining schemes are secure in both the past and the future time periods of the exposed key (forward and backward security). The PKI-based schemes [8-13] use public key certificates. So, there are communication cost and computational overhead of certificate management/verification especially in the data audit of multi-clients, which is the case of enterprise storage. These schemes still face the key-management problem. The Identity-based key-insulated scheme [15] is both forward and backward secure. In this scheme, the audit key is updated together by both the key updating server and the client. The secret key of the key updating server (we will call it update key for our reference) which is obtained from the PKG and which is used to update the audit key, remains unchanged in all the time periods. The scheme [15] assumes that this update key is secure. If both the update key and the audit key of

a time period are exposed, then the attacker will derive the audit keys of all the time periods and the auditing process will become a complete failure. How to safeguard the security of Identity-based cloud auditing if both the update key and the audit key are exposed, is still an open problem. Finding a solution for this problem is the primary objective of this work.

1.2 Literature Survey

Research works on key-exposure resilience are reviewed with a focus on the above said objective. In 2007, Ateniese et al. [7] defined a protocol for PDP and also proposed two PDP schemes to ascertain the possession of data in far-off servers. These schemes used homomorphic verifiable tags and the technique of random sampling. Following the scheme in [7], many PDP schemes [4 – 26] have evolved over time. But the key-exposure issue was first analysed only in 2015 by Yu et al. [8].

The PKI-based PDP schemes in [8-13] audit the data stored in a single cloud and are resilient to the exposure of the audit key. The scheme in [8] provides only forward security and so can prevent the forgery of authenticators of only past time periods. In this scheme, the client must update the key in every time period. To relieve the client from the burden of key updation, the scheme in [9] outsourced the updation of the encrypted version of the audit key to the TPA (Third Party Auditor). The scheme of [10] is both forward and backward secure. The update operation is jointly executed by both the TPA and the client. The update is done using the secret keys of both the TPA and the client. The scheme does not consider the exposure of the secret key of the TPA. The scheme in [11] handles this exposure by providing intrusion resilience. Here, the secret keys of both the TPA and the client also change in every time period. The adversary will not be able to obtain the audit key even if the secret keys of both the client and the TPA of different refreshing periods are exposed. The intrusion resilient scheme of [12] has the cloud server update the authenticators of data blocks using the key computed from the latest audit key. The forward secure scheme of [13] considers the leakage of partial information about the audit key. All these PKI-based schemes have computational and communication overhead of certificate usage [20]. Furthermore, the TPA faces the problem of key-management in the case of multiple clients.

Identity-based cloud auditing schemes of [4, 14-23] skip the need for certificates. In these schemes, the audit key for the initial time period is computed by a trusted third party, the Private Key Generator (PKG) using the identity ID of the client. The ID can be the client's e-mail address, phone number etc. The Identity-based key-exposure resilient schemes of [14, 15] are limited only to single cloud storage. The lattice-based scheme of [14] is only forward secure. The scheme of [15] provides both forward and backward security of the audit key. It also supports batch verification of multiple clients' data which is called batch auditing. It guards the privacy of data from the TPA. It is safe from semi-trusted cloud servers. A malicious cloud server can intentionally delete or modify the cloud data and can go unnoticed. Without retaining the original data blocks, the malicious cloud can generate valid proofs even without the audit key and can pass the audit. This limitation is a side-effect of the privacy-preserving feature. Moreover, as already mentioned, it does not consider the exposure of the update key.

The first Identity-based multicloud auditing scheme proposed by Wang [4] in 2015, preserves data privacy against the TPA but is not secure against malicious CSPs. Like in [15], the cloud can generate the audit response without the actual data blocks [16, 17]. This security flaw is fixed in [17]. The Identity-based multicloud schemes of [16-19] facilitate batch auditing.

From the analysis of PDP schemes with key-exposure resilience, it is observed that, PKI-based schemes incur certificate related overhead and still have the key-management problem. Some of the Identity-based PDP schemes are not secure against malicious CSPs. The exposure of the update key in Identity-based cloud auditing is still an open problem. Furthermore, multicloud auditing schemes have not regarded the key-exposure problem so far. These are some of the issues in the aspect of key-exposure open for further research.

1.3 Contribution

Based on the issues identified, in this research, an Identity-based PDP scheme with parallel key-insulation is proposed for auditing the data distributed to multiple clouds. The main features of the proposal are:

1. The system model of [15] is extended to support auditing of multiple cloud storage. A CSP interfacing between the TPA and the cloud servers must be a trusted entity. So, the cloud client/TPA in our scheme will interact directly with the cloud servers.
2. The proposed scheme extends the scheme of [15] to support parallel key-insulation [27]. With this feature, there is more than one key updating server (called the Key Updater) and one update key per Key Updater. In each time period, one of the update keys is used to update the audit key. The same update key will not be used in all the time periods. If only the audit key of a time period is exposed, then like the scheme in [15], our proposed scheme is both forward and backward secure. If one of the Key Updaters is compromised and the audit key of a time period is exposed to the adversary, then unlike in [15], the adversary cannot derive the audit keys of all the time periods. He can compute the audit key of only one time period.
3. The scheme protects the privacy of data using the hardness assumption of discrete-logarithm problem in cyclic groups. So, during audit, the TPA will not be able to fetch any information about the data from the response messages received from the clouds. It is also secure against malicious CSPs. The scheme in [15] is secure only against semi-trusted CSPs.
4. The scheme can be extended to handle batch auditing and can audit the data blocks of multiple clients simultaneously.

The remaining part of the paper is systematized as follows. The System Model, the notations, the Security Model and the Preliminaries of the proposed scheme are presented in Section 2. Section 3 gives the detailed algorithmic explanation of the proposed scheme followed by detailed security and performance analysis in Sections 4 and 5 respectively. Section 6 concludes highlighting the contributions and the future work.

2. Models and Notations

2.1. System Model

The System Model of [15] is extended to support parallel key-insulation and to audit the data distributed in multiple clouds. **Fig. 1** portrays our System Model. In this model, a firm is assumed to store business related data in various clouds. The lifetime of all data files in our system is divided into 0, 1, 2, 3, ... time periods. Like the scheme in [21], the system entity, PKG, is assumed to be from within the firm.
The role of each system entity is as follows:

- Cloud Servers are the cloud storage servers CS_1, CS_2, . . ., CS_J. There are many

CSPs and each storage server is assumed to be from a distinct cloud.

- Cloud Clients are the staff of the firm, authorized to utilize the cloud storage service for storing official data. A file F of a client is divided into data blocks of fixed size. In a time period t, the cloud client can generate the authentication tags for the data blocks. The data blocks and the corresponding tags are then distributed to the different clouds as per the information in the metadata. The metadata is a data storage dictionary. It contains one record per data block, giving the details of the cloud server where the data block is stored. It is prepared by the client and is shared with the TPA. The number of data blocks of a file can grow over time and the metadata can also be updated.
- Private Key Generator (PKG) is a server in the IT department of the firm and is considered to be a trusted party. It generates the secret keys called the update keys and the public keys for the Key Updaters. There is one update key per Key Updater. It generates the audit key for a cloud client only for the initial time period i.e., the zeroth time period.
- Key Updaters are the servers in the IT department and are presumed to be trusted. These servers are independent of each other. Each client has two secret keys, the time key and the audit key. A Key Updater computes the time key in time period $t \geq 1$ using the update key received from the PKG. The time key is then used by the client to update the audit key. The Key Updaters take turns in a round robin fashion to compute the time key of a client. In the above system model, there are three Key Updaters. In general, there can be any number of Key Updaters.

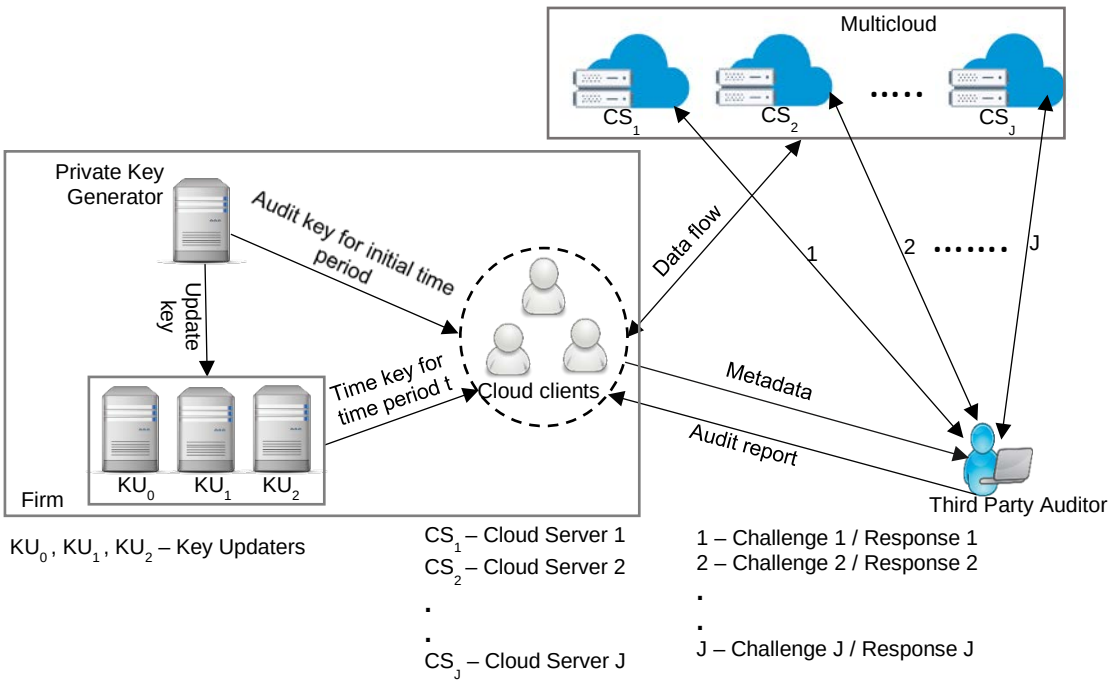


Fig. 1. The System Model

- TPA audits the data stored in multiple clouds. This entity is designed to handle auditing of single client's data as well as batch auditing. It possess the storage metadata and must be trusted. During audit, the TPA uses the metadata to generate

the challenge message for every cloud server. Each of these cloud servers reply back with a response message and the TPA verifies them all. After the audit, the TPA generates the audit report and sends it to the cloud client.

2.2. Notations

The notations used in the proposed scheme are presented in **Table 1**.

Table 1. Description of Notations

Notation	Description
G_1 and G_2	multiplicative cyclic groups of prime order q
Z_q^*	{1, 2, 3..., q - 1}
$e : G_1 \times G_1 \rightarrow G_2$	bilinear map used for pairing operation
g, u, P	generators of G_1
H_1, H_2, H_3	cryptographic hash functions
f	pseudo-random function
α	master secret key
PK	system public key
β_0, β_1 and β_2	update keys (secret keys) of Key Updaters KU_0, KU_1 and KU_2 respectively.
PK_0,PK_1 and PK_2	public keys of KU_0, KU_1 and KU_2 respectively
ID and t	identity and time period of a client
D_0	audit key of $t = 0$
D_t	audit key of $t \geq 1$
R_t	time key of $t \geq 1$
n	number of data blocks of a client
s	number of sectors of a data block
r	set of random numbers for data block sectors
CS_j	j^{th} cloud server
F	data file of a client
F_k	k^{th} data block of F, stored in CS_j
F_{kl}	a single sector of data block F_k
σ_k	authentication tag of data block F_k
F_{ik}	k^{th} data block of i^{th} client, stored in CS_j (used in batch auditing and in security proof)
σ_{ik}	authentication tag of data block F_{ik} (used in batch auditing and in security analysis)
MT	Metadata table of a client containing storage information on data blocks
C	set of challenged clouds
$chal$	an audit challenge message split into many $chal_j$
$chal_j$	a challenge message for CS_j
a_j	random key for CS_j
$\{v_k\}$	set of coefficients computed by CS_j using a_j
P_j	audit response for $chal_j$ from CS_j

2.3. Brief description of the algorithms and the Security Model

The algorithms of the Identity-based multicloud auditing protocol with parallel-key insulation are briefed as follows:

- **SysSetup** $(1^{\gamma}) \rightarrow (params, \alpha, \beta_0, \beta_1, \beta_2)$: The PKG runs this algorithm. The algorithm takes the security parameter γ and generates the system parameters – $params, \alpha, \beta_0, \beta_1$, and β_2. $params$ is a set of public parameters. α is the master secret key. $\beta_0, \beta_1, \beta_2$ are the secret keys of the Key Updaters KU_0, KU_1 and KU_2 respectively. These secret keys are also called the update keys. PK is the system public key. Similarly, PK_0, PK_1 and PK_2 are the public keys of KU_0, KU_1 and KU_2 respectively. These public keys are included in *params.*
- **InitialAuditKeyGen** $(params, \alpha, \beta_0, \beta_1, \beta_2, ID) \rightarrow (D_0)$: The PKG runs this algorithm. The algorithm creates the audit key D_0 of a client using $ID, params, \alpha, \beta_0, \beta_1$ and β_2. D_0 represents the audit key of time period $t = 0$.
- **TimeKeyGen** $(params, \beta_w, ID, t) \rightarrow (R_t)$: The Key Updater KU_w (KU_0 or KU_1 or KU_2) runs this algorithm at the start of every time period $t \geq 1$, to generate the time key R_t of a client. And it is created using the corresponding update key β_w. The inputs ID and t for this algorithm are received from the client. For a particular time period t, only one KU_w runs this algorithm.
- **AuditKeyUpdate** $(R_t, D_{t-1}) \rightarrow (D_t)$: The client runs this algorithm. The algorithm calculates the audit key D_t for the period t $(t \geq 1)$, using the audit key D_{t-1}. The time key of the current time period, R_t , is an input to this algorithm.
- **AuthGen** $(params, \{F_k\}, r, ID, t, D_t) \rightarrow (\sigma)$: It can be run by the client in any period *t*. This algorithm creates the set of authentication tags σ for the data blocks $\{F_k\}$ of file F using the audit key D_t . Here, F_k represents the k^{th} data block of *F*. The algorithm divides all the data blocks into same number of equal-sized sectors. r is a set of random numbers shared by the sectors of all the data blocks. For each data block, the client adds a record to the metadata table *MT*. The record contains the cloud server CS_j where the data block will be uploaded.
- **ChallengeGen** $(MT) \rightarrow (\{chal_j\})$: The TPA runs this algorithm to generate the audit challenge messages for multiple clouds. The algorithm first generates a single message called $chal$. $chal$ consists of random selection of indices of data blocks of a client signed in a particular time period *t* and an element $B \in G_2$. This element B is used to provide data privacy as well as to protect the cloud data from malicious CSPs. Using the metadata table *MT*, $chal$ is partitioned into a set of challenge messages $\{chal_j\}$. A $chal_j$ consists of the data block indices of a client stored in the same cloud server CS_j and *B*.
- **ProofGen** $(params, chal_j, \{F_k\}, \{\sigma_k\}, S) \rightarrow (P_j)$: This algorithm is run by the cloud servers in $\{CS_j\}$ who receive the challenge message $chal_j$ from the TPA. For the challenge $chal_j$, this algorithm generates the audit response P_j from the set of data blocks $\{F_k\}$ and the set of authenticators $\{\sigma_k\}$ (of a client) stored in CS_j .
- **ProofVerify** $(params, \{chal_j\}, \{P_j\}) \rightarrow (0/1)$: The algorithm verifies the cloud data of a client and is run by the TPA. The algorithm aggregates the responses $\{P_j\}$

received from all the challenged clouds in $\{CS_j\}$. It then verifies the aggregated response for a particular time period t(which is given in the challenge message) using the system public key PK and the public keys of the Key Updaters PK_0, PK_1 and PK_2. It returns 1, if all the challenged data blocks are secure in the multicloud storage.

BatchAuditProofGen and BatchAuditProofVerify are the algorithms for batch auditing.

- **BatchAuditProofGen** ($params$, $chal_j$, $\{F_{ik}\}$, $\{\sigma_{ik}\}$, $\{S_i\}$) → (P_j) : This algorithm is run by all the challenged clouds in $\{CS_j\}$. Unlike ProofGen, here $\{F_{ik}\}$ and $\{\sigma_{ik}\}$ refer to the set of data blocks and the set of authenticators of several clients (subscript i refers to the i^{th} client), stored in CS_j. S_i is unique for every file. And the audit challenge $chal_j$ consists of the indexes of data blocks of many cloud clients. The time period of the authentication tags of the data blocks of a client can be different from the other client. The algorithm generates the audit proof P_j for the data blocks of cloud clients in $chal_j$.
- **BatchAuditProofVerify** ($params$, $\{chal_j\}$, $\{P_j\}$) → (0/1) : This algorithm is similar to the ProofVerify algorithm. It is run by the TPA to verify the integrity of the data blocks of multiple clients. Unlike ProofVerify, the audit challenge message $chal_j$ and the audit proof message P_j involve the data blocks of multiple cloud clients.

Definition: An auditing protocol for multicloud storage is said to be resilient to the exposure of both the audit key and the update key, if for any adversary *A* (malicious CS), the probability with which *A* wins the Proof-Forge game on a set of data blocks of a file is negligible.

The Proof-Forge game is played by the adversary *A* and the challenger *C*. In this game, there are four phases, the Setup phase, the Query phase, the Challenge phase and the Forgery phase.

Setup phase: The challenger C sets up the public parameters $params$ and the public keys PK_0 ,PK_1 and PK_2 . These public parameters and the public keys are given to *A*. Let's assume that the update keys β_1 and β_2 are exposed to the adversary.

Query phase: In this phase, A can query the oracles H_1 , H_2 , H_3 , *AuditKey* and *AuthGen*. And C responds. Here, C runs the key update algorithms. In any time period t, A can get the audit key but must not ask the time key. A can compute the time key of certain time periods using the exposed keys β_1 and β_2 . In any time period t, A can query the authenticator of data blocks $\{F_k\}$ and C must compute and return the authenticators in this time period. As a time period ends, A can decide to either remain in this phase or go to the challenge phase.

Challenge phase: Here, C selects an identity ID^*, a time period t^* and some data block indices b_{1^*} , b_{2^*} , …., b_{c^*} and includes them in a challenge $chal^*$. C demands A to provide a valid proof of possession P^* for $chal^*$ such that

1. (ID^* , t^*), (ID^* , t^*+1) and (ID^* , t^*+2) were not used previously by A to query the *AuditKey* oracle.
2. t^* is the time period which requires β_0 to compute the time key R_{t^*} .
3. None of the b_{k^*} of $chal^*$ was used by A to query the *TagGen* oracle in time period t^*.

Adversary A would have obtained the audit keys which are earlier to t^* and/or which are later to t^*+2 in the query phase (here, 2 in t^*+ 2 is also *no_of_update_keys* - 1). A would also have computed the time keys of t^*- 2 , t^*- 1 , t^*+ 1 and t^*+ 2 using the exposed keys β_1 and

β_2. Even then, it is practically not easy for *A* to calculate the audit key of t^* without the time key R_{t^*} of t^*. Computation of R_{t^*} requires the update key β_0 which is not known to A.

Forgery phase: Here, *A* responds with a proof P^* for $chal^*$. A is said to win the game only if ProofVerify ($chal^*$, P^*, PK, PK_0, PK_1, PK_2) = 1.

2.4. Preliminaries

The proposed scheme uses pairing-based cryptographic algorithms to implement auditing. This section presents the bilinear map used by these algorithms and the computational problem used in security analysis.

- **Bilinear Map**
  Consider G_1 and G_2 to be two multiplicative cyclic groups of large prime order *q*. The map $e : G_1 \times G_1 \rightarrow G_2$ is a bilinear map [28] if all these axioms are satisfied:
  - *Bilinearity:* For all *u*, *v* ϵ G_1 and *x*, *y* ϵ Z_q^*, $e(u^x, v^y) = e(u, v)^{xy}$
  - *Non-degeneracy:* There exists $g, h \in G_1$ such that $e(g, h) = 1$
  - *Computability:* For all $u, v \in G_1$, we can compute $e(u, v) \in G_2$ in polynomial time.

- **CDH problem**
  Given (g, g^a, g^b), where g is a generator of multiplicative group G_1 of prime order q and $a, b \in Z_q^*$, the Computational Diffie-Hellman (CDH) problem [28] is to compute g^{ab}. It is assumed that the CDH problem in G_1 is hard i.e., there is no efficient algorithm to solve the problem with non-negligible probability.

3. Detailed Description of our Proposed Scheme

An Identity-based scheme with parallel key-insulation is proposed for multicloud auditing. It uses the System Model and the bilinear map concept of section 2. It extends the key updation technique of the general signature scheme in [29] to support three update keys. It also supports blockless verification. A data file *F* of a client has many data blocks distributed in cloud servers of different CSPs. Each block is divided into a fixed number of equal-sized sectors.

The lifetime of the audit key D_t of a client is partitioned into time periods 0, 1, 2, 3, 4, ... and the value of the key changes in each period. 0 is the initial time period (i.e., the time period $t = 0$ initially). 1 is the next time period. 2 comes after 1. 3 comes after 2. Likewise, the time period t increments by 1. The audit key is updated at the beginning of each period. In any period, new data blocks of the file *F* are signed using the audit key of the corresponding period and then stored in the cloud servers. It is the client who decides which block is to be stored in which server (metadata). Similarly, in any period, a block stored in any cloud server can be modified and re-signed using the audit key of the corresponding period. The client, if he intends can also re-sign all the stored data blocks using the latest audit key without any modification. Unlike the audit key, the public keys used for integrity verification will not change in any period.

Unlike in [15], there are three Key Updaters KU_0, KU_1 and KU_2. Each of the Key Updaters obtain an update key from the PKG. β_0, β_1 and β_2 are the update keys of KU_0, KU_1 and KU_2 respectively. These update keys are independent of each other. At $t = 0$, the audit key D_0 is generated by the PKG using the master secret key α and all the update keys β_0, β_1 and β_2 (**Fig. 2**). In every $t \geq 1$, the audit key D_{t-1} gets updated to D_t using the time key R_t. After the updation, both D_{t-1} and R_t are deleted.

The Key Updaters take turns in a round robin fashion to generate the time key R_t. The Key Updater KU_1 uses β_1 to create the time key in time periods 1, 4, 7, 10, 13.... KU_2 creates the time key in time periods 2, 5, 8, 11, 14... using β_2. Similarly, KU_0 generates the time key in time periods 3, 6, 9, 12, 15..... using β_0. In general, in time period t, KU_w computes the time key R_t using the update key β_w. Here, $w = t \, mod \, 3$. In all $t \geq 1$, the audit key D_t is computed using the time key R_t of t and the audit key D_{t-1} of t - 1. This is called updation of the audit key**. Fig. 3** illustrates the updation of the audit key in time periods 1, 2 and 3. Similarly, **Fig. 4** illustrates the updation in time periods 4, 5 and 6.

As per our proposal, if an update key and an audit key of a period are both exposed to the adversary, then he can compute the audit key of only one time period. For example, if both β_0 and D_{t-1} are exposed, then the adversary can compute D_t. Similarly, if both β_0 and D_t are exposed, then the adversary can derive D_{t-1}. But he cannot obtain the audit keys of the remaining time periods.

During audit, the TPA prepares a challenge message consisting of random data block indices, splits the challenge message into a set of smaller challenge messages using the metadata and distributes them to the corresponding clouds. Here, the TPA can challenge all the clouds or some of the clouds (where the challenged blocks are present). The responses from the challenged clouds are then aggregated and verified. To provide security against malicious cloud service providers, B must be different in each audit, i.e., the element $\rho \in Z_q^*$ used to generate B must differ in each audit.

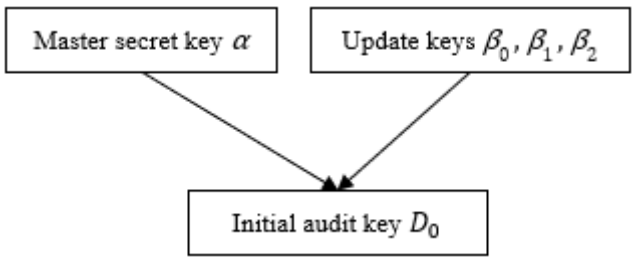
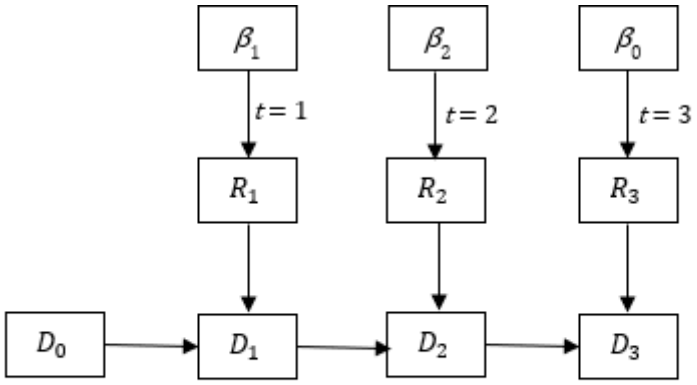


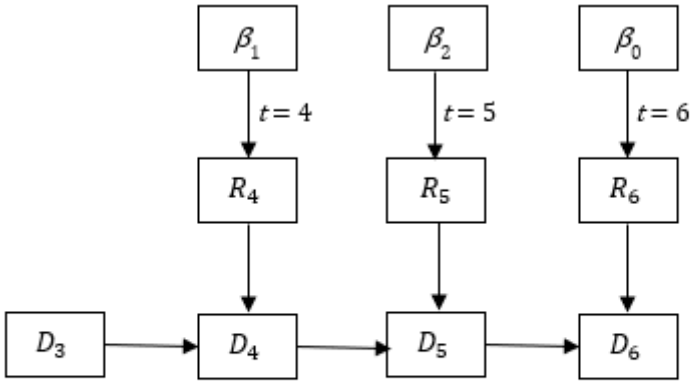
Fig. 2. Generation of initial audit key



R_1, R_2, R_3 – Time keys of time periods 1, 2 and 3 respectively

D_0, D_1, D_2, D_3 – Audit keys of time periods 0, 1, 2 and 3 respectively

Fig. 3. Updation of the audit key in time periods 1, 2 and 3



R_4, R_5, R_6 – Time keys of time periods 4, 5 and 6 respectively

D_3, D_4, D_5, D_6 – Audit keys of time periods 3, 4, 5 and 6 respectively

Fig. 4. Updation of the audit key in time periods 4, 5 and 6

Each algorithm of the proposed scheme is listed in steps as:

- **SysSetup**
  1. Given a security parameter γ, select two multiplicative cyclic groups, G_1 and G_2 of prime order $q > 2^{\gamma}$ and a bilinear map $e: G_1 \times G_1 \rightarrow G_2$.
  2. Select three generators $g, u, P \in G_1$.
  3. Randomly choose four elements – $\alpha, \beta_0, \beta_1, \beta_2 \in Z_q^*$. Let α be the master secret key. Let β_0, β_1 and β_2 be the update keys (secret keys) of the Key Updaters KU_0, KU_1 and KU_2 respectively.
  4. Compute the system public key using the master secret key as

$$PK = g^{\alpha} \tag{1}$$

5. PK_0 , PK_1 and PK_2 are the public keys of the Key Updaters KU_0, KU_1 and KU_2 respectively. Compute

$$PK_0 = g^{\beta_0} \quad (2)$$
$$PK_1 = g^{\beta_1} \quad (3)$$
$$PK_2 = g^{\beta_2} \quad (4)$$

6. Select three cryptographic hash functions H_1 , $H_2 : \{0, 1\}^* \rightarrow G_1$ and $H_3 : \{0, 1\}^* \times G_1 \rightarrow Z_q^*$.
7. Choose the pseudo-random function f: $Z_q^* \times \{1, 2, 3, ..., n\} \rightarrow Z_q^*$.
8. Include the public parameters in *params* as *params* = (G_1 , G_2 , e , g , u , P, PK , PK_0 , PK_1 , PK_2 , H_1 , H_2 , H_3 , f) and publish them. Send β_0 , β_1 and β_2 secretly to the respective Key Updaters.

- **InitialAuditKeyGen**
  1. Input the identity $ID \in \{0, 1\}^*$ of a client. Compute the audit key $D_0 \in G_1$ for the initial time period $t = 0$ using the secret keys α , β_0 , β_1 and β_2 as,

$$D_0 = H_1(ID)^{\alpha} \, H_2(ID \,||\, -2)^{\beta_1} \, H_2(ID \,||\, -1)^{\beta_2} \, H_2(ID \,||\, 0)^{\beta_0} \quad (5)$$

  2. Send D_0 to the client over a protected channel.

The client receives the audit key and checks its validity using the equation

$$e(g \, , D_0) = e\big(PK \, , H_1(ID)\big) \, e\big(PK_1 \, , H_2(ID \,||\, -2)\big) \, e\big(PK_2 \, , H_2(ID \,||\, -1)\big) \, e\big(PK_0 \, , H_2(ID \,||\, 0)\big) \quad (6)$$

- **TimeKeyGen**
  1. Compute the time key $R_t \in G_1$ for the client for the time period t using the secret key β_w as,

$$R_t = (H_2(ID \,||\, t) \, (H_2(ID \,||\, t - 3))^{-1})^{\beta_w} \quad (7)$$

  Here, t is the current period and $(t-3)$ is the time period when the time key was last updated by the same Key Updater KU_w. β_w is the update key of Key Updater KU_w and $w = t \bmod 3$. Depending on the value of t, one of β_0 , β_1, β_2 is used in (7).
  2. Send R_t to the client over a protected channel.

- **AuditKeyUpdate**
  1. Validate the time key R_t received from the Key Updater as

$$e(g \, , \, R_t) = e(PK_w \, , H_2(ID \,||\, t) \, (H_2(ID \,||\, t - 3))^{-1}) \quad (8)$$

  Here, PK_w is the public key of the Key Updater KU_w which was used to generate R_t.
  2. Update the audit key using,

$$D_t = R_t \, D_{t-1} \quad (9)$$

  3. Delete the keys R_t and D_{t-1}.

- **AuthGen**

Randomly select $x \in Z_q^*$ and compute $S = g^x$. Let F_k be the k^{th} data block of the file F of the cloud client, predetermined to be stored in cloud server CS_j. Here, k is

the data block index. Partition each data block F_k into s number of sectors, F_k= $\{F_{k1}, F_{k2}, ..., F_{ks}\}$. For each sector l of a data block, there is a random number r_l and the set $r = \{r_l \mid r_l \in Z_q^*, l = 1, 2, 3, ..., s\}$ is common to all the data blocks. For each data block F_k,

1. Compute the hash value, $h_k = H_3(ID_i \,||\, name \,||\, CS_j || \, k \,||\, t\,, S)$ where $name$ is the name of the file.
2. Compute $M_k = \sum_{l=1}^{s} r_l F_{kl} \in Z_q^*$ (10)
3. Compute $T_k = P^{h_k x} \, u^{M_k} \, D_t$ (11)
   Here, D_t is the audit key of the current time period t. The authentication tag of data block F_k is $\sigma_k = \{T_k, t\}$. Add the record $\Phi_k = (name, k, CS_j, t)$ to the metadata table MT. The set of authentication tags of F is $\sigma = \{S, \{\sigma_k\}\}$. S is the same for all the data blocks of F.

The client uploads the block/tag pairs to the respective cloud servers and sends the metadata table *MT* to the TPA. The TPA uses the metadata table to generate the audit challenge.

- **ChallengeGen**
  1. Select c number of data block indexes of a data file of a client randomly. These data blocks must have been signed in a particular time period *t*.
  2. Select an element ρ, randomly from Z_q^* and compute $B \in G_2$ as,
     $$B = e(u, g^{\rho}) \quad (12)$$
  3. Generate the challenge message, $chal = (I, t, A, B)$ where $I = I_1 \cup I_2 \ldots \cup I_j \ ... \cup I_J$. Here, I_j is the index set for cloud server CS_j and $I_j = \{b_1, b_2, b_3, \ldots\}$ where $b_1, b_2, b_3, ...$ are the indexes of data blocks stored in CS_j. $A = \{a_j \mid j \in C \text{ and } a_j \in Z_q^*\}$ where C is the set of challenged clouds in $\{CS_j\}$ and a_j is the key for CS_j. a_j will be used by CS_j as an input to pseudo-random function in ProofGen.
  4. Split the challenge message $chal$ into multiple challenge tokens $chal_j = (I_j, t, a_j, B)$. There is one $chal_j$ per cloud server CS_j.
  5. Send each challenge message $chal_j$ to the respective CS_j.

- **ProofGen**
  1. On receiving the challenge message $chal_j$, compute the coefficient $v_k \in Z_q^*$ for each block index *k* in I_j of $chal_j$ using the pseudo-random function as
     $$v_k = f_{a_j}(k) \quad (13)$$
  2. Compute $\check{T}_j \in G_1$ using the authenticators T_k of blocks indexed in $chal_j$ and its corresponding coefficients v_k computed in step 1 as
     $$\check{T}_j = \prod_{k \in I_j} (T_k)^{v_k} \quad (14)$$
  3. For each sector *l* of the data blocks, compute $f = \sum_{k \in I_j} v_k F_{kl} \in Z_q^*$ and set
     $$B_l = B^f \quad (15)$$
  4. Send the proof $P_j = \{S, \check{T}_j, \{B_l\}_{l=1,2,3,...,s}, t\}$ to the TPA. Here, s is the number of sectors.

- **ProofVerify**
  1. After receiving the audit proof messages $\{P_j\}$ from all the challenged cloud servers, compute $\check{\mathrm{T}} = \prod_{j \in C} \check{\mathrm{T}}_j$ (16)
     Here C is the set of challenged cloud servers in $\{CS_j\}$.
  2. Compute $N = \prod_{j \in C} \prod_{l=1}^{s} B_l{}^{r_l}$ (17)
  3. Verify the integrity of the data stored in multiple clouds using the equation,
$$\begin{aligned}\left(g^{\rho}, \check{\mathrm{T}}\right) = e\left(S^{\rho}, P^{\sum_{k \in I}\left(h_k v_k\right)}\right) e\left(PK^{\rho}, H_1(ID)^{\sum_{k \in I} v_k}\right) \\ e\left(PK_{w'''}^{\rho}, H_2(ID \,||\, t-2)^{\sum_{k \in I} v_k}\right) \\ e\left(PK_{w''}^{\rho}, H_2(ID \,||\, t-1)^{\sum_{k \in I} v_k}\right) \\ e\left(PK_{w'}^{\rho}, H_2(ID \,||\, t)^{\sum_{k \in I} v_k}\right) N\end{aligned} \quad (18)$$
     Here, $h_k = H_3\left(ID_i \,||\, name \,||\, CS_j || \, k \,||\, t, S\right)$, $I = I_1 \cup I_2 \ldots \cup I_j \ldots \cup I_J$, $w' = t \bmod 3$, $w'' = (t-1) mod 3$ and $w''' = (t-2) mod 3$. If any of w' or w'' or w''' is negative, then the constant 3 is added to it.
  4. Return 1 if L.H.S. = R.H.S. in (18), otherwise return 0.

Proof of Correctness for (18):

$$e\left(g^{\rho}, \check{\mathrm{T}}\right) = e\left(g^{\rho}, \prod_{j \in C} \check{\mathrm{T}}_j\right) \quad (19)$$

substituting (14) in (19)

$$e\left(g^{\rho}, \check{\mathrm{T}}\right) = e\left(g^{\rho}, \prod_{j \in C} \prod_{k \in I_j} (T_k)^{v_k}\right) \quad (20)$$

substituting (11) in (20)

$$\begin{aligned} e\left(g^{\rho}, \check{\mathrm{T}}\right) &= e\left(g^{\rho}, \prod_{j \in C} \prod_{k \in I_j} \left(P^{h_k x} u^{M_k} D_t\right)^{v_k}\right) \\ &= e\left(g^{\rho}, \prod_{k \in I} \left(P^{h_k x}\right)^{v_k}\right) e\left(g^{\rho}, \prod_{k \in I} (u^{M_k})^{v_k}\right) e\left(g^{\rho}, \prod_{k \in I} (D_t)^{v_k}\right) \end{aligned} \quad (21)$$

substituting (9) in (21)

$$\begin{aligned} e\left(g^{\rho}, \check{\mathrm{T}}\right) &= e\left((g^x)^{\rho}, \prod_{k \in I} P^{h_k v_k}\right) e\left(g^{\rho}, u^{\sum_{k \in I}(M_k v_k)}\right) e\left(g^{\rho}, \prod_{k \in I} (R_t D_{t-1})^{v_k}\right) \\ &= e\left(S^{\rho}, P^{\sum_{k \in I}(h_k v_k)}\right) B^{\sum_{k \in I}(M_k v_k)} \\ &\quad e\left(g^{\rho}, \prod_{k \in I} \left(H_2(ID \,||\, t)^{\beta'_w} \left(H_2(ID \,||\, t-3)^{\beta'_w}\right)^{-1} R_{t-1} D_{t-2}\right)^{v_k}\right) \\ &= e\left(S^{\rho}, P^{\sum_{k \in I}(h_k v_k)}\right) N\, e\left(g^{\rho}, \prod_{k \in I} \left(H_2(ID \,||\, t)^{\beta'_w} H_2(ID \,||\, t-1)^{\beta''_w} H_2(ID \,||\, t-2)^{\beta'''_w} \left(H_2(ID \,||\, t-6)^{\beta'_w}\right)^{-1} \left(H_2(ID \,||\, t-7)^{\beta''_w}\right)^{-1} \left(H_2(ID \,||\, t-8)^{\beta'''_w}\right)^{-1} R_{t-6} D_{t-5}\right)^{v_k}\right) \end{aligned}$$

$$\begin{aligned} e\left(g^{\rho}, \check{\mathrm{T}}\right) = e\left(S^{\rho}, P^{\sum_{k \in I}(h_k v_k)}\right) N\, e\left((g^{\alpha})^{\rho}, \prod_{k \in I} H_1(ID)^{v_k}\right) \\ e\left(\left(g^{\beta'''_w}\right)^{\rho}, \prod_{k \in I} H_2(ID \,||\, t-2)^{v_k}\right) \\ e\left(\left(g^{\beta''_w}\right)^{\rho}, \prod_{k \in I} H_2(ID \,||\, t-1)^{v_k}\right) \\ e\left(\left(g^{\beta'_w}\right)^{\rho}, \prod_{k \in I} H_2(ID \,||\, t)^{v_k}\right) \end{aligned}$$

$$\begin{aligned} e\left(g^{\rho}, \check{\mathrm{T}}\right) = e\left(S^{\rho}, P^{\sum_{k \in I}(h_k v_k)}\right) e\left(PK^{\rho}, H_1(ID)^{\sum_{k \in I} v_k}\right) \\ e\left(PK_{w'''}^{\rho}, H_2(ID \,||\, t-2)^{\sum_{k \in I} v_k}\right) e\left(PK_{w''}^{\rho}, H_2(ID \,||\, t-1)^{\sum_{k \in I} v_k}\right) \\ e\left(PK_{w'}^{\rho}, H_2(ID \,||\, t)^{\sum_{k \in I} v_k}\right) N \end{aligned}$$

Here, $w' = t \bmod 3, w'' = (t-1) mod 3$ and $w''' = (t-2) mod 3$

For the purpose of batch auditing, the challenge is altered to include the block indices of multiple cloud clients - $chal_j = (I_j, a_j, B)$. Here, $I_j = \{(i, t, b_1, b_2, b_3, \dots)\}$. The time period of the data blocks of a client indexed in $chal_j$ may be different from other clients.

- **BatchAuditProofGen**
  1. After receiving the challenge message $chal_j$, generate the coefficient for each index *k* in the challenge $chal_j$ using (13) .
  2. Compute $\check{T}_j = \prod_{i,k \in I_j} (T_{ik})^{v_k} \in G_1$ (22)
  3. For each sector *l* of client *i*, compute
  $$f = \sum_{i,k \in I_j} v_k F_{ikl} \in Z_q^* \quad (23)$$
  $$B_{il} = B^f \quad (24)$$
  Totally there will be $number\_of\_clients \times s$ values of $B_{il}$where s is the number of sectors.
  4. Send the proof $P_j = \left\{\{S_i\}_{i \in I_j}, \check{T}_j, \{B_{il}\}_{i \in I_j, l = 1,2,3,\dots,s}, \{t\}_{i \in I_j}\right\}$ to the TPA.

- **BatchAuditProofVerify**
  1. Compute $\check{T} = \prod_{j \in C} \check{T}_j$ (25)
  2. Compute N_i value for each client *i* whose data blocks are involved in the challenge as $N_i = \prod_{j \in C} \prod_{l=1}^{s} B_{il}^{r_l}$ (26)
  3. Verify the data of multiple clients stored in multiple clouds using the equation,
  $$\begin{aligned}(g^\rho, \check{T}) = \prod_{i \in O} e\left(S_i^\rho, P^{\sum_{k \in I}(h_k v_k)}\right) e\left(PK^\rho, \prod_{i \in O} H_1(ID_i)^{\sum_{k \in I} v_k}\right) \\ e\left(PK_0^\rho, \prod_{i \in O} H_2(ID_i \,||\, t_a)^{\sum_{k \in I} v_k}\right) \\ e\left(PK_1^\rho, \prod_{i \in O} H_2(ID_i \,||\, t_b)^{\sum_{k \in I} v_k}\right) \\ e\left(PK_2^\rho, \prod_{i \in O} H_2(ID_i \,||\, t_c)^{\sum_{k \in I} v_k}\right) \prod_{i \in O} N_i \quad (27)\end{aligned}$$
  Here, $h_k = H_3(ID_i \,||\, name \,||\, CS_j \,||\, k \,||\, t, S_i)$, one of t_a , t_b , $t_c$is time period t which is given in the challenge and the remaining are time periods t - 1 and t - 2. The notations t_a , t_b , t_c are used in the above equation since the time period of data blocks may differ for different clients. Here, I is the union of data block indices of multiple cloud servers and $I = I_1 \cup I_2 \dots \cup I_j \dots \cup I_J$ where I_j is the set of data block indices of cloud server $\{CS_j\}$. O is the set of cloud clients i.e., $O = \{i\}$ whose data is to be verified in batch auditing.
  4. Return 1 if L.H.S. = R.H.S. in (27), otherwise return 0.

4. Security Analysis

Theorem 1 (Resilience to key-exposure): If the CDH problem in G_1 is hard, then the proposed multicloud auditing scheme is resilient to the exposure of both the audit key and the update key.

Proof: The CDH problem is to find g^{ab}, given (g, g^a, g^b) where g is a generator of multiplicative group G_1 of order q and $a, b \in Z_q^*$. Let A be an adversary and L be a challenger. The goal of L is to solve the CDH problem. Algorithm C simulates the challenger. A Proof-Forge game is used between A and C. If A can produce a valid proof with non-negligible

probability, then A is said to have won the game and therefore C can solve the CDH problem.

- **SysSetup phase:** Consider G_1 and G_2 to be multiplicative cyclic groups of order q. C sets the system public key, $PK = g^{\alpha} \in G_1$ and the public keys of the Key Updaters, $PK_0 = g^{a}$, $PK_1 = g^{\beta_1}$ and $PK_2 = g^{\beta_2}$. C picks m and n randomly from Z_q^*, computes $u = g^m \in G_1$ and $P = g^n \in G_1$. C then forwards (G_1, G_2, e, g, u, P, PK, PK_0, PK_1, PK_2) to A.

- **Query phase:** C maintains a H_1 list, a H_2 list, a H_3 list, an *AuditKey* list and a *Tag* list which are initially empty. Adversary A can query the H_1 *Oracle*, the H_2 *Oracle*, the H_3 *Oracle*, the *AuditKey Oracle* or the *AuthGen Oracle* at any time. Consider that A makes 'X' AuditKey queries and 'Y' AuthGen queries. Let i denote a client. For an AuditKey query or an AuthGen query on identity ID_i of a client i, let us assume that a H_1 query and three H_2 queries (for t, t-1 and t-2) were already been made on that identity.

  - H_1 *Oracle Query*: When A asks for H_1 output with identity input ID_i, C looks for an entry for ID_i in the H_1 list and finds the tuple (ID_i, x_i, g^{x_i}) and returns g^{x_i}. If ID_i is not in the list, then C picks $x_i \in Z_q^*$ at random, sets $H_1(ID_i) = g^{x_i}$, adds the tuple (ID_i, x_i, g^{x_i}) to the H_1 list and returns g^{x_i}.

  - H_2 *Oracle Query:* The adversary A queries the H_2 *Oracle* with input (ID_i, t). C returns $h_{2i} \in G_1$ from the H_2 list, if the given input matches the tuple $(ID_i, t, d_i, y_{i,t}, h_{2i})$ in the list. Otherwise, C flips a coin $d_i \in \{0, 1\}$ such that, the probability $P[d_i = 0] = \lambda$ and the probability $P[d_i = 1] = 1 - \lambda$.
    i. If $d_i = 0$, C randomly selects $y_{i,t} \in Z_q^*$ and computes
    $$H_2(ID_i \,||\, t) = g^{y_{i,t}} = h_{2i}$$
    ii. If $d_i = 1$, C randomly selects $y_{i,t} \in Z_q^*$ and computes
    $$H_2(ID_i \,||\, t) = (g^{b})^{y_{i,t}} = h_{2i}$$
    In both the scenarios, C puts the tuple $(ID_i, t, d_i, y_{i,t}, h_{2i})$ to the H_2 list and returns h_{2i}.

  - H_3 *Oracle Query:* For A's input $(ID_i, name, CS_j, k, t, S_i)$, C returns the value z_i from the H_3 list, if $(ID_i, name, CS_j, k, t, S_i, z_i)$ is already in the list. If there is no entry, then C randomly picks z_i from Z_q^*, returns z_i and adds the tuple (ID_i, *name*, CS_j, k, t, S_i, z_i) to the H_3 list.
  - *AuditKey Oracle Query*: When asked the audit key of (ID_i, t), C accesses the tuple $(ID_i, t, D_{i,t})$ in the *AuditKey* list and gives $D_{i,t}$ to A. If the tuple is absent, then C finds the tuple $(ID_i, t, d_i, y_{i,t}, h_{2i})$ in the H_2 list. Here,
    i. If $d_i = 0$, C computes the audit key of time period t as $D_{i,t} = (g^{x_i})^{\alpha} (g^{y_{i,t-2}})^{\beta_w'''} (g^{y_{i,t-1}})^{\beta_w''} (g^{y_{i,t}})^{\beta_w'}$. Here, $w' = t \bmod 3$, $w'' = (t-1) \bmod 3$ and $w''' = (t-2) \bmod 3$. C adds $(ID_i, t, D_{i,t})$ to the *AuditKey* list and gives $D_{i,t}$ to A.

ii. If $d_i = 1$, C aborts with an unsuccessful message.

- *AuthGen Oracle Query:* A queries the authentication tag of (ID_i, t, F_{ik}) and C returns σ_{ik} if the tuple $(ID_i, t, F_{ik}, \sigma_{ik})$ is available in the *Tag* list. Otherwise, C recovers the tuple $(ID_i, t, d_i, y_{i,t}, h_{2i})$ from the H_2 list. Here,
  i. If $d_i = 0$, C picks p_i from Z_q^* and sets $S_i = g^{p_i}$. Picks up $z_i \in Z_q^*$ and adds the tuple $(ID_i, name, CS_j, k, t, S_i, z_i)$ to the H_3 list. It then computes
  $T_{ik} = (g^n)^{z_i p_i} (g^m)^{M_{ik}} \quad (g^{x_i})^{\alpha} (g^{y_{i,t-2}})^{\beta'''_w} (g^{y_{i,t-1}})^{\beta''_w} (g^{y_{i,t}})^{\beta'_w}$
  where
  $M_{ik} = \sum_{l=1}^{s} r_l F_{ikl}$, $w' = t \bmod 3$, $w'' = (t-1) \bmod 3$ and $w''' = (t-2) \bmod 3$. Sets $\sigma_{ik} = \{S_i, T_{ik}, t\}$ and adds $(ID_i, t, F_{ik}, \sigma_{ik})$ to the Tag list. Also returns σ_{ik} to A.
  ii. If $d_i = 1$, C aborts reporting failure.

- **Challenge phase:** Here, C selects an identity ID^* and challenges A with $chal^* = \{b_{1^*}, b_{2^*}, \ldots, b_{c^*}, t^*, a_j, B\}$. Here, $b_{1^*}, b_{2^*}, \ldots, b_{c^*}$ is the set of indexes I of data blocks stored in CS_j. Value $B = e(u, g^{\rho^*}) \in G_2$ where ρ^* is a random element of Z_q^*.

- **Forgery phase:** Here, A responds with a valid proof $P^* = \{S^*, \check{T}^*, \{B_l^*\}_{l=1,2,3,\ldots,s}, t^*\}$ with a probability as least ϵ.
  - *Breaking CDH:* Algorithm C accesses the tuple $(ID^*, t^*, d^*, y_{t^*}, h_{2^*})$ in the H_2 list and checks the value of d^*.
    i. If $d^* = 0$, C aborts.
    ii. If $d^* = 1$, C proceeds as follows:

$$e(g^{\rho^*}, \check{T}^*) = e\left((S^*)^{\rho^*}, P^{\sum_{k^* \in I}(h_{k^*} v_{k^*})}\right) e\left(PK^{\rho^*}, H_1(ID^*)^{\sum_{k^* \in I}(v_{k^*})}\right) e\left(PK_1^{\rho^*}, H_2(ID^* || t^* - 2)^{\sum_{k^* \in I}(v_{k^*})}\right) e\left(PK_2^{\rho^*}, H_2(ID^* || t^* - 1)^{\sum_{k^* \in I}(v_{k^*})}\right) e\left(PK_0^{\rho^*}, H_2(ID^* || t^*)^{\sum_{k^* \in I}(v_{k^*})}\right) N^* \quad (28)$$

where $h_{k^*} = H_3(ID^* || name^* || CS_j || k^* || t^*, S^*)$

substituting the following in (28),

P by g^n, h_{k^*} by z_{k^*}, PK by g^{α}, PK_0 by g^{a}, PK_1 by g^{β_1}, PK_2 by g^{β_2}, $H_1(ID^*)$ by g^{x^*}, $H_2(ID^* || t^* - 2)$ by $g^{y_{t^*-2}}$, $H_2(ID^* || t^* - 1)$ by $g^{y_{t^*-1}}$, $H_2(ID^* || t^*)$ by $(g^b)^{y^*}$ and N^* by $e(g^{\rho^*}, (g^m)^{\sum_{k^* \in I}(M_{k^*} v_{k^*})})$ where g^m is u.

$$e(g^{\rho^*}, \check{T}^*) = e\left((S^*)^{\rho^*}, (g^n)^{\sum_{k^* \in I}(z_{k^*} v_{k^*})}\right) e\left((g^{\alpha})^{\rho^*}, (g^{x^*})^{\sum_{k^* \in I}(v_{k^*})}\right) e\left((g^{\beta_1})^{\rho^*}, (g^{y_{t^*-2}})^{\sum_{k^* \in I}(v_{k^*})}\right) e\left((g^{\beta_2})^{\rho^*}, (g^{y_{t^*-1}})^{\sum_{k^* \in I}(v_{k^*})}\right) e\left((g^{a})^{\rho^*}, (g^{b})^{y^* \sum_{k^* \in I}(v_{k^*})}\right) e\left(g^{\rho^*}, (g^m)^{\sum_{k^* \in I}(M_{k^*} v_{k^*})}\right)$$

$$\check{T}^* = (S^*)^{n\sum_{k^*\epsilon I}(z_{k^*}v_{k^*})}\left(g^{\alpha x^*}\right)^{\sum_{k^*\epsilon I}(v_{k^*})}\left(g^{\beta_1}\right)^{y_{t^*-2}\sum_{k^*\epsilon I}(v_{k^*})}$$
$$\left(g^{\beta_2}\right)^{y_{t^*-1}\sum_{k^*\epsilon I}(v_{k^*})}\left(g^{ab}\right)^{y^*\sum_{k^*\epsilon I}(v_{k^*})}\ g^{m\sum_{k^*\epsilon I}(M_{k^*}v_{k^*})}$$
$$\check{T}^* = (S^*)^{n\sum_{k^*\epsilon I}(z_{k^*}v_{k^*})}\left(PK^{x^*}\right)^{\sum_{k^*\epsilon I}(v_{k^*})}PK_1{}^{y_{t^*-2}\sum_{k^*\epsilon I}(v_{k^*})}$$
$$PK_2{}^{y_{t^*-1}\sum_{k^*\epsilon I}(v_{k^*})}\left(g^{ab}\right)^{y^*\sum_{k^*\epsilon I}(v_{k^*})}\ g^{m\sum_{k^*\epsilon I}(M_{k^*}v_{k^*})}$$

Let V be a variable introduced to find the value of g^{ab},

$$V = \check{T}^*(S^*)^{-n\sum_{k^*\epsilon I}(z_{k^*}v_{k^*})}PK^{-x^*\sum_{k^*\epsilon I}(v_{k^*})}PK_1{}^{-y_{t^*-2}\sum_{k^*\epsilon I}(v_{k^*})}$$
$$PK_2{}^{-y_{t^*-1}\sum_{k^*\epsilon I}(v_{k^*})}\ g^{-m\sum_{k^*\epsilon I}(M_{k^*}v_{k^*})}$$
$$g^{ab} = V^{\left(y^*\sum_{k^*\epsilon I}(v_{k^*})\right)^{-1}}$$

Therefore, C solves the CDH problem with a non-negligible probability. The following events are used in the calculation of probability of C's success:

1. E_1: C does not abort in the AuditKey query
2. E_2: C does not abort in the AuthGen query
3. E_3: A generates a valid proof of possession
4. E_4: Event E_4 and $d^* = 1$ for the tuple $(ID^*, t^*, d^*, y_{t^*}, h_{2^*})$ in the H_2 list

$$P[E_1 \wedge E_2 \wedge E_3 \wedge E_4] = \lambda^X \lambda^Y \varepsilon(1-\lambda)$$
$$= \varepsilon \frac{(X+Y)^{X+Y}}{(X+Y+1)^{X+Y+1}}$$

Therefore, g^{ab} can be computed and this implies that the CDH assumption is not true. So, we can say that it is not practical to produce a valid proof of possession. Hence, it is finalized that the proposed multicloud auditing scheme is resilient to the exposure of both the audit key and the update key.

Theorem 2 (Error detectability): Let n block-tag pairs be stored and c out of n block-tag pairs be challenged. If s out of n block-tag pairs are modified or deleted, then the error detection probability P_y satisfies the equation:

$$1 - \left(\frac{n-s}{n}\right)^c \leq P_y \leq 1 - \left(\frac{n-c+1-s}{n-c+1}\right)^c \quad (29)$$

Proof: Let Y be a discrete random variable representing the number of corrupted/deleted blocks in the challenged block list (at least one block in the challenge must match the modified/deleted block). So,

$$P_y = \mathrm{P}(Y \geq 1)$$
$$= 1 - (Y = 0)$$
$$= 1 - \left(\frac{n-s}{n}\right)\left(\frac{n-1-s}{n-1}\right)\left(\frac{n-2-s}{n-2}\right)\ldots\ldots\left(\frac{n-c+1-s}{n-c+1}\right)$$

Therefore, $1 - \left(\frac{n-s}{n}\right)^c \leq P_y \leq 1 - \left(\frac{n-c+1-s}{n-c+1}\right)^c$

When c = 300 and s = 1% of n, the probability of error detection is atleast 95%.

5. Performance Analysis

In **Table 2**, the proposed scheme is compared with three existing Identity-based schemes, Lan et al.'s scheme [17], Zhao et al.'s scheme [18] and Nithya et al.'s scheme [15] for different features. Like the proposed scheme, Nithya et al.'s scheme is resilient to the exposure of the

audit key (the other two schemes are not resilient to key-exposure). So, for the time of updation of the audit key, the proposed scheme is compared with Nithya et al.'s scheme [15]. For the remaining parameters such as size of audit messages, tag generation time, proof verification time etc. the proposed scheme is compared with both Lan et al.'s scheme and Zhao et al.'s scheme since they audit multicloud data whereas Nithya et al.'s scheme does not.

Table 2. Comparison of the features of Identity-based schemes

	Lan et al.'s scheme	**Zhao et al.'s scheme**	**Nithya et al.'s scheme**	**Proposed scheme**
Audits multicloud data	Yes	Yes	No	Yes
Resilient to the exposure of the audit key	No	No	Yes	Yes
Resilient to the exposure of the update key	No	No	No	Yes
Data Privacy against the TPA	Yes	No	Yes	Yes
Security against the data deletion/modification attack of the malicious CSP	Yes	Yes	No	Yes
Batch auditing	Yes	Yes	Yes	Yes

5.1 Analysis of the size of data stored in clouds and the size of audit messages

Let F be the data file of a cloud client. The data blocks of the file are distributed to the clouds. The total amount of cloud space required for storing all the data blocks of file F and the corresponding authentication tags is estimated and analysed. **Table 3** compares the size of storage and the size of the audit messages between the three schemes. The meaning of the notations used in **Table 3**, are as follows:

$|F|$ - length of the data file F
$|G_1|$ - length of an element of group G_1
$|G_2|$ - length of an element of group G_2
$|Z_q^*|$ - length of an element of group Z_q^*
n - number of data blocks of F
s - number of sectors of a data block
$|i|$ - length of a data block index
c - number of challenged blocks
m - number of challenged clouds
d - number of clients
$|t|$ - length of a time period

In **Table 3**, the extra overhead $n \mid t \mid$ in the data storage of the proposed scheme is due to the key-resilience feature and $\mid t \mid$ is negligible compared to $\mid G_1 \mid$ and $\mid Z_q^* \mid$. In **Table 3**, the size of the audit challenge represents the total size of all the challenge messages (which are sent from the TPA to the m clouds). The challenge message of Zhao et al.'s scheme is the smallest. This is because Zhao et al.'s scheme is not privacy-preserving (**Table 2**) whereas both the proposed scheme and Lan et al.'s scheme support data privacy. Compared to Lan et al.'s scheme, the challenge message is smaller in the proposed scheme. This is because to preserve data privacy, the proposed scheme generates only one element of G_2 for every challenged cloud but Lan et al.'s scheme generates $d \times s$ elements of G_2 for every challenged cloud. Moreover, in Lan et al.'s scheme, a cloud server called the combiner acts as a mediator between the TPA and the challenged clouds during audit, increasing the cost of communication still further. In the table, $(2 \mid Z_q^* \mid + d s \mid G_2 \mid)$ *bytes* are sent from the TPA to the combiner.

The size of the response message represents the total size of all the audit response messages (which are sent from the m clouds to the TPA). It is the smallest for Zhao et al.'s scheme ($\mid Z_q^* \mid << \mid G_2 \mid$) since it does not preserve data privacy (**Table 2**). Compared to Lan et al.'s scheme, the size of the response message is greater for the proposed scheme by ($(m - 1) \mid G_1 \mid - \mid G_2 \mid$) *bytes*. The reasons are,

- In Lan et al.'s scheme, $\mid G_1 \mid + \mid G_2 \mid$ *bytes* are sent from the combiner to the TPA
- In the proposed scheme, every cloud server includes one more element of G_1 from the authenticator list

5.2 Analysis of Computation Time

Table 4 compares the computation time between the three schemes. In the table, T_H, T_e, T_p and T_c represent the time to compute a hash to a point in the group G_1, exponentiation in G_1, pairing from G_1 to G_2 and exponentiation in G_2 respectively. The remaining operations such as inverse in the group G_1, multiplication in G_1, multiplication in the group G_2 and operations in Z_q^* contribute negligible computation time and are therefore omitted. The proposed scheme, Lan et al.'s scheme and Zhao et al.'s scheme are simulated using the *C* programming language with the Pairing-Based Cryptography (PBC) Library version 0.5.14 [30] in the following environment:

- OS: Ubuntu Linux 17.04
- CPU: Intel(R) Core(TM) i5 CPU 650 @ 3.20 GHz
- Physical Memory: 4.00 GB RAM

Type 'A' pairing is used for testing. So, $\mid G_1 \mid = 128$ *bytes*, $\mid G_2 \mid = 128$ *bytes* and $\mid Z_q^* \mid = 20$ *bytes*. $\mid F \mid = 2$ GB and the file is divided into 1,000,000 data blocks. Each data block is 2 KB in size and is divided into 100 sectors (i.e., $s = 100$) of equal size.

The time of updating the audit key is the time taken together by both the TimeKeyGen and the AuditKeyUpdate algorithms. As seen in **Fig. 5**, the time in the proposed scheme and Nithya et al.'s scheme is similar except for $t = 0$. For $t = 0$, the proposed scheme requires two more hash to a point in G_1 and two more exponentiation operations ($2\ T_H + 2\ T_e$) than Nithya et al.'s scheme. This is due to the resilience feature of the proposed scheme to the exposure of both the audit key as well as the update key (**Table 2**). Whereas Nithya et al.'s scheme collapses in the case of exposure of both the keys.

Table 3. Comparison of the data storage size and the size of audit messages

Schemes	Data storage on the cloud servers (*bytes*)	Audit challenge (*bytes*)	Audit response (*bytes*)
Lan et al.'s scheme	$\lvert F \rvert + n \lvert G_1 \rvert$	$m \lvert Z_q^* \rvert + c \lvert i \rvert + m\ d\ s \lvert G_2 \rvert + 2 \lvert Z_q^* \rvert + d\ s \lvert G_2 \rvert$	$m(\lvert G_1 \rvert + d\ s \lvert G_2 \rvert) + \lvert G_1 \rvert + \lvert G_2 \rvert$
Zhao et al.'s scheme	$\lvert F \rvert + (n+1) \lvert G_1 \rvert$	$m \lvert Z_q^* \rvert + c \lvert i \rvert$	$m(2 \lvert G_1 \rvert + d\ s \lvert Z_q^* \rvert)$
Proposed scheme	$\lvert F \rvert + (n+1) \lvert G_1 \rvert + n \lvert t \rvert$	$m \lvert Z_q^* \rvert + c \lvert i \rvert + m \lvert G_2 \rvert + m\ d \lvert t \rvert$	$m(2 \lvert G_1 \rvert + d\ s \lvert G_2 \rvert + d \lvert t \rvert)$

Table 4. Comparison of computation time.

	Lan et al.'s scheme	Zhao et al.'s scheme	Proposed scheme
Tag Generation	$n\mathrm{T_H} + n(s+1)\mathrm{T_e}$	$n\mathrm{T_H} + (2n+1)\mathrm{T_e}$	$(2n+1)\mathrm{T_e}$
Challenge Generation	$2\ d\mathrm{T_e} + d\ s\mathrm{T_p}$	----------	$\mathrm{T_e} + \mathrm{T_p}$
Proof Generation	$c\mathrm{T_e} + (m\ d\ s)\ \mathrm{T_c}$	$c\mathrm{T_e}$	$c\mathrm{T_e} + (m\ d\ s)\ \mathrm{T_c}$
Proof Verification	$c\mathrm{T_H} + (c+1)\mathrm{T_e} + (d+1)\mathrm{T_p}$	$(c+d)\mathrm{T_H} + (c+d)\mathrm{T_e} + (d+2)\mathrm{T_p}$	$4d\mathrm{T_H} + (6d+5)\mathrm{T_e} + (d+5)\mathrm{T_p} + sd\ \mathrm{T_c}$

It can be seen from **Fig. 6**, that the tag generation time of the proposed scheme is lesser compared to the existing schemes since its *AuthGen* algorithm replaces the costly hash to an element of G_1 by hash to an element of Z_q^*. Comparatively, the time is much higher for Lan et al.'s scheme due to greater number of exponentiations (in **Table 4**, $n(s+1)\mathrm{T_e} > (2n+1)\mathrm{T_e}$).

Let c be the number of challenged blocks. **Fig. 7** and **Fig. 8** depict the time taken to generate the audit challenge messages for $m = 10$ *clouds* with $d = 1$ *client* and $d \geq 50$ *clients* respectively. Zhao et al.'s scheme is not privacy-preserving and so its challenge generation algorithm does not contain any of the major operations (mentioned above). Hence, the time consumed by Zhao et al.'s scheme is very minimal and is therefore omitted in **Table 4** and in the figures. In both the figures, the time taken by the proposed scheme is constant and is much lesser than Lan et al.'s scheme. This is because, the proposed scheme requires lesser number of exponentiations and pairing operations ($\mathrm{T_e}$ and $\mathrm{T_p}$) than Lan et al.'s scheme (**Table 4**).

Fig. 9 and **Fig. 10** illustrate the total time taken by $m = 10$ *clouds* to generate the audit response messages with $d = 1$ *client* and $d \geq 50$ *clients* respectively. Here, the proof generation time of Lan et al.'s scheme and the proposed scheme are almost similar and also higher than Zhao et al.'s scheme. This is also due to the same reason that Zhao et al.'s scheme is not privacy-preserving.

Fig. 11 and **Fig. 12** portray the time taken to verify the audit proof messages of 10 *clouds* (m) with $d = 1$ *client* and $d \geq 50$ *clients* respectively. In both the figures, the proof verification time of the proposed scheme is much lesser than the other two schemes due to lesser number of hash to an element of G_1 and exponentiations. ($c >> d$, $\mathrm{T_c} < \mathrm{T_p}$, $\mathrm{T_c} < \mathrm{T_e}$ and $\mathrm{T_c} < \mathrm{T_H}$ in **Table 4**).

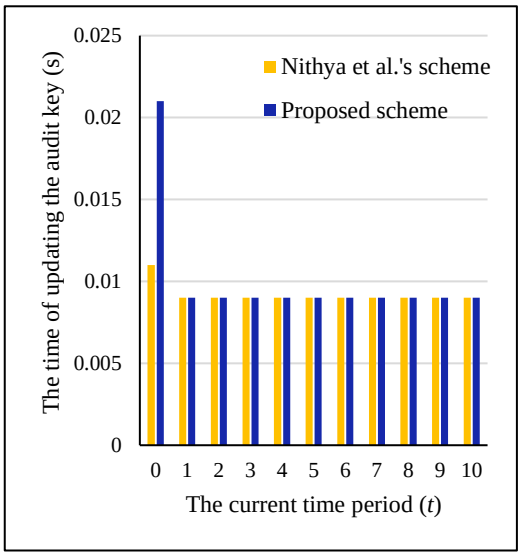


Fig. 5. The time of updating the audit key

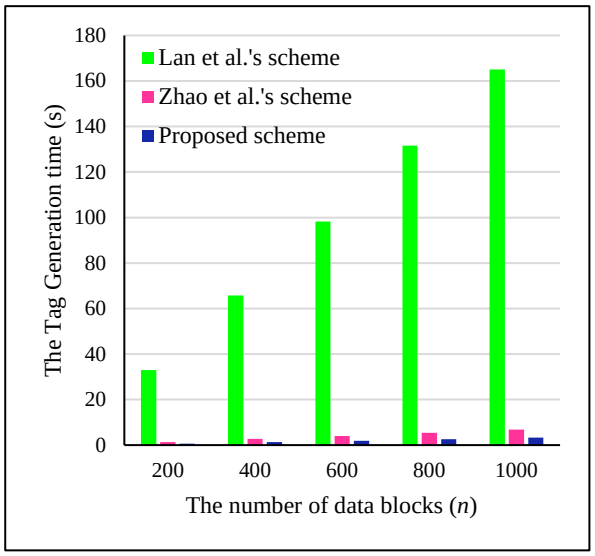


Fig. 6. The Tag Generation time

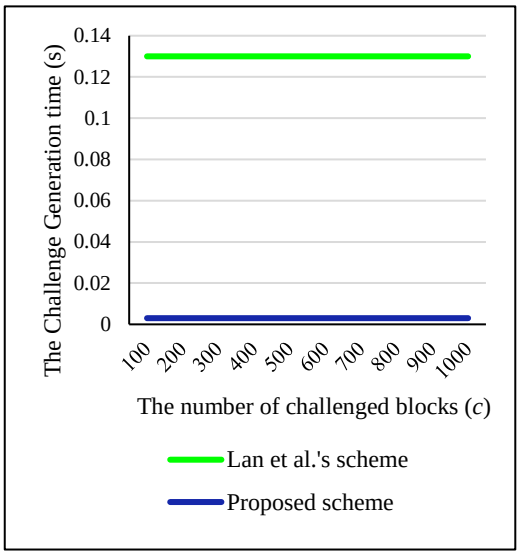


Fig. 7. The Challenge Generation time. Here 10 clouds (i.e., $m = 10$) are challenged.

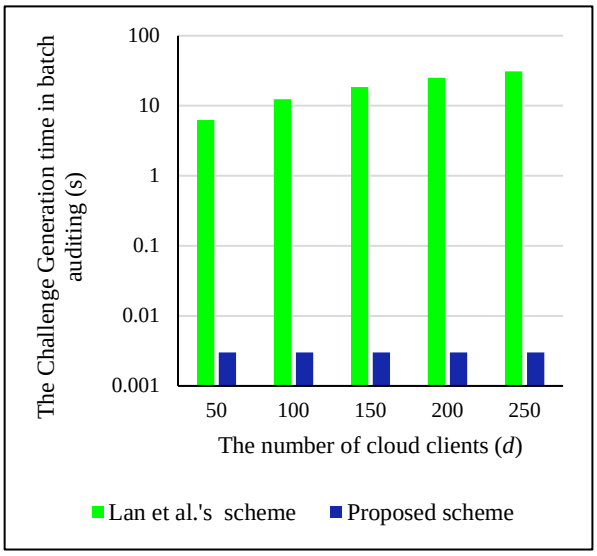


Fig. 8. The Challenge Generation time in batch auditing. Here, 10 clouds are challenged with $c = d \times 100$ data blocks.

6. Conclusion and Future Work

In this paper, an Identity-based PDP scheme with parallel key-insulation is proposed for multicloud storage. To our knowledge, it is the first key-exposure resilient scheme in distributed cloud storage auditing. It reduces the risk of exposure of update keys. The update key is a secret key used to update the auditing secret key (audit key). Our scheme will not fail, even if some of the update keys and some of the audit keys are both exposed. It protects the privacy of data during third party auditing and is also secure against malicious Cloud Service Providers. The scheme is extended further, to handle auditing delegations from multiple clients using a batch approach.

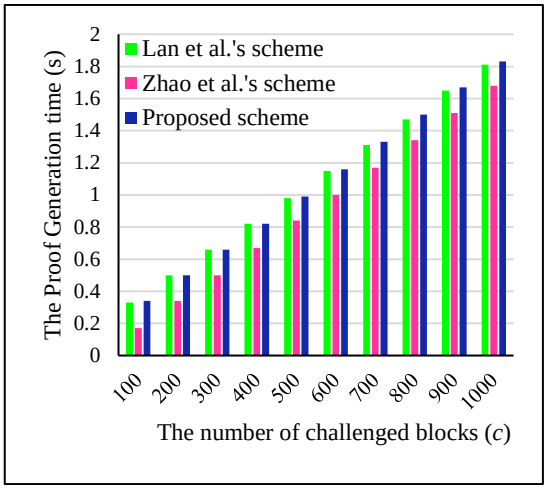


Fig. 9. The Proof Generation time of 10 clouds

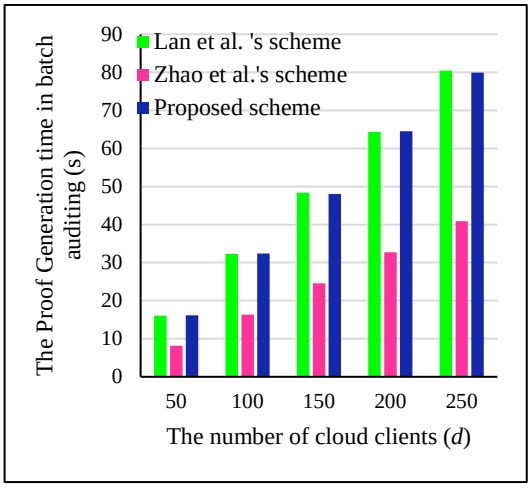


Fig. 10. The Proof Generation time of 10 clouds in batch auditing. Here, $c = d \times 100$ data blocks.

Both the numerical and the experimental analyses show that the proposed scheme can efficiently verify the integrity of cloud data. Although the proof generation time of the proposed scheme and Lan et al.'s scheme are similar, our proposed scheme has the added advantage of being resilient to key-
exposure whereas Lan et al.'s scheme is not.

Our future work is to extend the above scheme to provide intrusion resilience i.e., change the update key in every time period. This is to strengthen the security of auditing even if many of the update keys and many of the audit keys are both exposed.

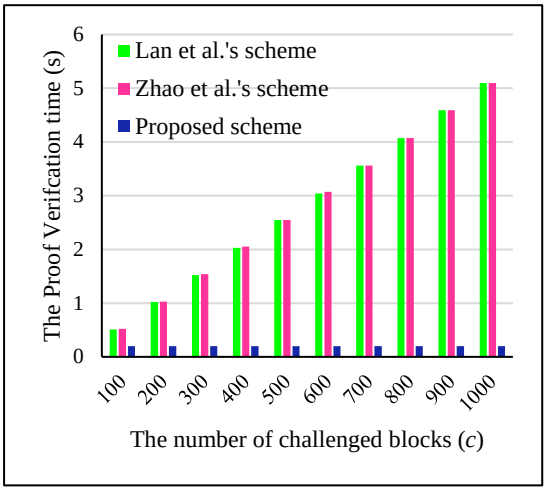


Fig. 11. The Proof Verification time. The audit proof from 10 clouds are verified.

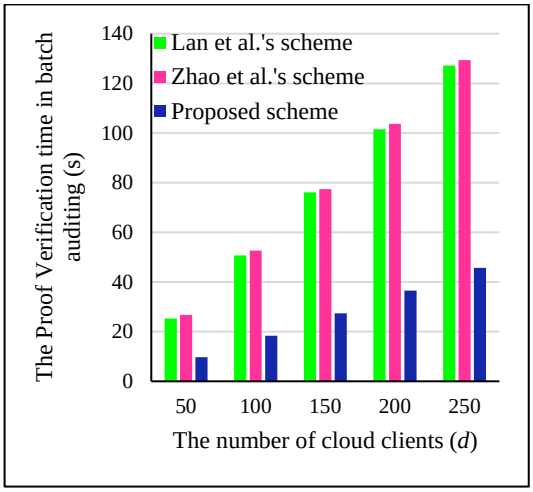


Fig. 12. The Proof Verification time in batch auditing. Here, $m = 10$ clouds and $c = d \times 100$ data blocks.

Acknowledgments

We would like to sincerely thank Dr.S. Jayaprakash, Emeritus Professor, formerly with IIT Madras and Late Dr.V. Sankaranarayanan, Former Director (University Project), B.S.A. University for their advice and encouragement which were very helpful in improving the quality of this work.

References

[1] B. Borcherding, "Why Enterprises are Accelerating Cloud Adoption," *Forbes Technology Council*, 2020. Article (CrossRef Link)

[2] K. Weins, "Cloud Computing Trends:2021 State of the Cloud Report," *Flexera*, 2021. Article (CrossRef Link)

[3] Forrester Consulting, "86% of enterprises have adopted a multi-cloud strategy," *Help Net Security*, 2018. Article (CrossRef Link)

[4] H. Wang, "Identity-Based Distributed Provable Data Possession in Multicloud Storage," *IEEE Trans. on Services Computing*, vol. 8, no. 2, pp. 328-340, Mar. 2015. Article (CrossRef Link)

[5] Y. Zhu, G. Ahn, H. Hu, and S. Yau, "Dynamic audit services for outsourced storages in clouds," *IEEE Trans. on Services Computing*, vol. 6, no. 2, pp. 227-238, Apr. 2013. Article (CrossRef Link)

[6] H. Wang, "Proxy provable data possession in public clouds," *IEEE Trans. on Services Computing*, vol. 6, no. 4, pp. 551-559, Oct. 2013. Article (CrossRef Link)

[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 Conf. on Computer and Communs. Security(CCS'07)*, pp. 598-609, Oct. 2007. Article (CrossRef Link)

[8] J. Yu, K. Ren, C. Wang, and V. Varadharajan, "Enabling cloud storage auditing with key-exposure resistance," *IEEE Trans. on Information Forensics & Security*, vol. 10, no. 6, pp. 1167-1179, June, 2015. Article (CrossRef Link)

[9] J. Yu, K. Ren, and C. Wang, "Enabling cloud storage auditing with verifiable outsourcing of key updates," *IEEE Trans. on Information Forensics & Security*, vol. 11, no. 6, pp. 1362–1375, June, 2016. Article (CrossRef Link)

[10] J. Yu, and H. Wang, "Strong key-exposure resilient auditing for secure cloud storage," *IEEE Trans. on Information Forensics and Security*, vol. 12, no. 8, pp. 1931–1940, Aug. 2017. Article (CrossRef Link)

[11] R. Ding, Y. Xu, J. Cui, and H. Zhong, "A public auditing protocol for cloud storage system with Intrusion Resilience," *IEEE Systems Journal*, vol. 14, no. 1, pp. 633-644, Mar. 2020. Article (CrossRef Link)

[12] Y. Xu, S. Sun, J. Cui, and H. Zhong, "Intrusion-resilient public cloud auditing scheme with Authenticator update," *Elsevier, Information Sciences*, vol. 512, pp. 616–628, Feb. 2020. Article (CrossRef Link)

[13] C. Hu, Y. Xu, P. Liu, J. Yu, S. Guo and M. Zhao, "Enabling cloud storage auditing with key-exposure resilience under continual key-leakage," *Elsevier, Information Sciences*, vol. 520, pp. 15–30, May, 2020. Article (CrossRef Link)

[14] X. Zhang, H. Wang, and C. Xu, "Identity-based key-exposure resilient cloud storage public auditing scheme from lattices," *Elsevier, Information Sciences*, vol. 472, pp. 223–234, Jan. 2019. Article (CrossRef Link)

[15] S.M.V. Nithya and R. Uthariaraj, "Identity-based Public Auditing Scheme for Cloud Storage with Strong Key-Exposure Resilience," *Hindawi Security & Commun. Networks*, pp. 1–13, Jan. 2020. Art. No. 4838497. Article (CrossRef Link)

[16] S. Peng, F. Zhou, J. Xu, and Z. Xu, "Comments on "Identity-Based Distributed Provable Data Possession in Multicloud Storage"," *IEEE Trans. on Services Computing*, vol. 9, no. 6, pp. 996-998, Nov. 2016. Article (CrossRef Link)

[17] C. Lan, H. Li, and C. Wang, "Analysis of the comments on "Identity-Based Distributed Provable Data Possession in Multicloud Storage"," *IEEE Trans. on Services Computing*, vol. 14, no. 1, pp. 44-46, Jan. 2021. Article (CrossRef Link)
[18] J. Zhao, C. Xu, and Kefei Chen, "A Security-Enhanced Identity-Based Batch Provable Data Possession Scheme for Big Data Storage," *KSII Transactions on Internet and Information Systems*, vol. 12, no. 9, pp. 4576-4598, Sep. 2018. Article (CrossRef Link)
[19] F. Zhou, S. Peng, J. Xu, and Z. Xu, "Identity-Based Batch Provable Data Possession with Detailed Analyses," *International Journal of Foundations of Computer Science*, vol. 28, no. 6, pp. 743-760, 2017. Article (CrossRef Link)
[20] H. Wang, Q. Wu, B. Qin, and J. Domingo-Ferrer, "Identity-based remote data possession checking in public clouds," *IET Information Security*, vol. 8, no. 2, pp. 114-121, Mar. 2014. Article (CrossRef Link)
[21] Y. Wang, Q. Wu, B. Qin, W. Shi, R.H. Deng, and J. Hu, "Identity-based Data outsourcing with comprehensive auditing in clouds," *IEEE Trans. on Information Forensics & Security*, vol. 12, no. 4, pp. 940-952, Apr. 2017. Article (CrossRef Link)
[22] Y. Zhang, J. Yu, R. Hao, C. Wang and K. Ren, "Enabling Efficient User Revocation in Identity-Based cloud storage auditing for shared big data," *IEEE Trans. on Dependable & Secure computing*, vol. 17, no. 3, pp. 608-619, May 2020. Article (CrossRef Link)
[23] Y. Li, Y. Yu, G. Min, W. Susilo, J. Ni and K. R. Choo, "Fuzzy Identity-Based Data Integrity Auditing for Reliable Cloud Storage Systems," *IEEE Trans. on Dependable & Secure computing*, vol. 16, no. 1, pp. 72-83, Jan. 2019. Article (CrossRef Link)
[24] Y. Zhu, H. Hu, G. Ahn, and M. Yu, "Cooperative provable data possession for integrity verification in multicloud storage," *IEEE Trans. on Parallel & Distributed Systems*, vol. 23, no. 12, pp. 2231-2244, Dec. 2012. Article (CrossRef Link)
[25] C. Wang, S.S. M Chow, Q. Wang, K. Ren and W. Lou, "Privacy-preserving public auditing for Secure cloud storage," *IEEE Trans. on Computers*, vol. 62, no. 2, pp. 362-375, Feb. 2013. Article (CrossRef Link)
[26] B. Wang, B. Li, and H. Li, "Oruta: Privacy-preserving public auditing for shared data in the cloud," *IEEE Trans. on Cloud Computing*, vol. 2, no. 1, pp. 43-56, Jan. 2014. Article (CrossRef Link)
[27] G. Hanaoka, Y. Hanaoka and H. Imai, "Parallel Key-Insulated Public Key Encryption," in *Proc. of the 9th Springer Int. Conf. on Theory and Practice in Public Key Cryptography (PKC 2006)*, LNCS 3958, pp. 105-122, 2006. Article (CrossRef Link)
[28] D. Boneh, "A brief look at pairings based cryptography," in *Proc. of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS '07)*, pp. 19-26, Oct. 2007. Article (CrossRef Link)
[29] R. B. Amarapu and P.V. Reddy, "Efficient Identity-Based Parallel Key-Insulated Signature Scheme using Pairings over Elliptic Curves," *Journal of Scientific & Industrial Research*, vol. 77, pp. 24-28, Jan. 2018. Article (CrossRef Link)
[30] B. Lynn, "The Pairing-Based Cryptographic Library," 2018. Article (CrossRef Link)

Dr. V. Rhymend Uthariaraj is currently Professor and Dean of Online and Distance Learning at Crescent University, Chennai, India. His previous positions were in Anna University, Chennai as the Professor and the Director of Ramanujan Computing Center, the Secretary of TNEA, the Head of the Department of Information Technology, the Head of the Department of Media Sciences and the Director of Knowledge Data Centre. He has been the governing council member at C-DAC, MHRD from 2010-2014. He has guided 25 Ph.D. scholars and has published more than 100 papers in various International Journals and reputed Conferences. His research interests include Pervasive Computing, High Performance Networks, Network Security, Optimization, Cloud Computing and Internet of Things.

S. Mary V. Nithya is currently a Full Time Research Scholar of Ramanujan Computing Centre, Anna University, Chennai. She received her B.E. degree in Computer Science from Madurai Kamaraj University in 2003 and her M.Tech. degree in Information Technology from Anna University in 2013. She has got 7 years of working experience in IT industries like Wipro Technologies, Bangalore and Aricent Technologies, Chennai. Her research interests include Cloud Computing and Security.