Hindawi
Security and Communication Networks
Volume 2020, Article ID 4838497, 13 pages
https://doi.org/10.1155/2020/4838497

WILEY | Hindawi

Research Article

Identity-Based Public Auditing Scheme for Cloud Storage with Strong Key-Exposure Resilience

S. Mary Virgil Nithya and V. Rhymend Uthariaraj

Ramanujan Computing Centre, Anna University, Guindy, Chennai 600 025, Tamil Nadu, India

Correspondence should be addressed to S. Mary Virgil Nithya; nithya.roshan08@gmail.com

Received 10 June 2019; Revised 27 September 2019; Accepted 30 October 2019; Published 27 January 2020

Academic Editor: David Megias

Secured storage system is a critical component in cloud computing. Cloud clients use cloud auditing schemes to verify the integrity of data stored in the cloud. But with the exposure of the auditing secret key to the Cloud Service Provider, cloud auditing becomes unsuccessful, however strong the auditing schemes may be. Therefore, it is essential to prevent the exposure of auditing secret keys, and even if it happens, it is necessary to minimize the damage caused. The existing cloud auditing schemes that are strongly resilient to key exposure are based on Public Key Infrastructure and so have challenges of certificate management/verification. These schemes also incur high computation time during integrity verification of the data blocks. The Identity-based schemes eliminate the usage of certificates but limit the damage due to key exposure, only in time periods earlier to the time period of the exposed key. Some of the key exposure resilient schemes do not provide support for batch auditing. In this paper, an Identity-based Provable Data Possession scheme is proposed. It protects the security of Identity-based cloud storage auditing in time periods both earlier and later to the time period of the exposed key. It also provides support for batch auditing. Analysis shows that the proposed scheme is resistant to the replace attack of the Cloud Service Provider, preserves the data privacy against the Third Party Auditor, and can efficiently verify the correctness of data.

1. Introduction

Cloud storage is one of the services provided by the cloud, where a client can store data in the data centre managed by the Cloud Service Provider (CSP). To deal with the hardware, software, and maintenance challenges of data growth and its associated costs, many enterprises are moving their business-critical data and applications to the cloud [1–4]. There is an increased adoption of public cloud [5] mainly to reduce the cloud cost. Even though the CSPs guarantee that the stored data will be secure and intact, many issues affecting the confidentiality, integrity, availability, etc. of the stored information [6] are likely to happen in the cloud. Among them, integrity is considered to be critical as the cloud client does not have physical control over stored data. For the same reason, traditional integrity checking mechanism cannot be directly applied to cloud storage. Downloading all the data and verifying them periodically would result in high communication costs and time. Therefore, checking the integrity of the cloud data remotely is truly a security challenge for cloud clients.

1.1. Issues in Cloud Storage Auditing. Data integrity is the overall completeness, accuracy, and consistency of data, i.e., the data stored in the cloud is not altered or deleted without the client's knowledge. Ateniese et al.'s auditing model for Provable Data Possession (PDP) at untrusted stores [7] is widely used for cloud data integrity checking. In this model, the cloud client divides a data file into blocks, generates an authentication tag for each block using a secret key called the auditing secret key, and then uploads the data blocks and the authentication tags to the cloud. During audit, a challenge-response protocol is used between the client and the CSP, wherein the client verifies a small number of random set of data blocks without downloading them. This type of verification is called blockless verification. This feature facilitates the client to outsource the data auditing to a Third Party

Auditor (TPA), and this type of auditing is called public auditing.

Most of the PDP-based auditing schemes use Public Key Infrastructure- (PKI-) based cryptography, Identity-based cryptography, etc. PKI-based auditing systems [10–18] have communication and computational complexities of certificate management and certificate verification [19]. Identity-based auditing schemes [19–25] improve the efficiency by eliminating the use of certificates but have some limitations such as key exposure, computational overhead of dynamic data updates, etc. Further, some of these schemes do not provide support for simultaneous auditing of cloud data of multiple clients which is called batch auditing. Batch auditing reduces the computation time of the TPA in auditing the data of multiple clients.

Exposure of the auditing secret key of a cloud client is a serious problem since the integrity of data cannot be assured. There are many ways in which the auditing secret key is exposed. Improper key management can lead to key exposure [28]. Even a semitrusted CSP may try to conceal the data loss events caused by Byzantine failures etc. from the clients [28] in order not to lose business. The CSP may attempt to steal the auditing secret key of the client, generate fake data, and also valid authentication tags that may go unidentified during audit. Some cloud clients may update the data in the cloud and regenerate the data block tags using mobile devices [29] that have resource constraints. These devices are not secure and are therefore prone to key exposure. Once the auditing secret key is exposed, all the data block tags, irrespective of whether they were generated before or after the exposure become invalid, resulting in loss of data integrity.

Most often, key exposure is unknown and also difficult to detect. Therefore, it is necessary to be proactive in minimizing the damage caused by key exposure even before knowing that such an exposure has happened. Minimizing the damage caused by auditing secret key exposure means that the attacker is unable to forge any of the data block tags, even after acquiring the key. The existing Identity-based cloud auditing scheme [26] tackles the auditing secret key-exposure problem with the forward security property [27] where the secret key evolves over time. With forward security, the authenticators generated using the secret keys of time periods earlier to the time period of the exposed secret key could be preserved. The authenticators of remaining time periods, i.e., the authenticators generated using the exposed key and those generated from the secret keys of time periods later to the time period of the exposed key could be forged [15]. This is because the adversary can derive the future auditing secret keys from the exposed one. Considering these facts, this research focuses on minimizing the impact of damage to the data integrity in time periods both earlier and later to the time period of the exposed key in Identity-based cloud storage auditing schemes.

1.2. Literature Survey. Some of the research works on the issues of auditing secret key exposure in cloud storage auditing and minimizing the problems due to the key-exposure are reviewed in this section.

Provable Data Possession (PDP) [7] and Proofs of Retrievability (PoR) [8, 9] are the two models used to check the integrity of data outsourced to untrusted and remote servers without actually downloading the data. These models use the technique of random sampling to audit the data. The PoR model uses error correcting codes to recover corrupted portions of the data file. The cloud auditing schemes are either based on PDP or PoR. PDP is the widely used model [10–26, 28, 29].

The Public Key Infrastructure- (PKI-) based cloud auditing schemes in [15–17, 28] provide key-exposure resilience. The scheme in [28] is forward secure. The auditing secret key of the schemes given in [15, 16] is jointly updated by both the client and the TPA, and it is both forward and backward secure. So, only the authenticators which were generated using the exposed secret key can be forged. The intrusion resilient scheme in [16] divides a time period into several refreshing periods and refreshes the secret values (that are used to update the auditing secret key) in each refreshing period. Therefore, the adversary can compute the auditing secret key of the client only if the client and the TPA are compromised in the same refreshing period. The scheme in [17] encrypts the auditing secret key and outsources the updation of the encrypted key to the TPA. The schemes in [16, 17, 28] use a binary tree for key updation. Therefore, these schemes do not support unbounded time periods and the key update time is not constant, it varies depending on the location of the node (associated with the current time period) in the tree. The number of exponentiation operations in the ProofVerify algorithm of the schemes in [17, 28] is dependent on the bit length of the node corresponding to the time period of integrity verification. The key update time in the scheme in [15] is constant, and it supports unbounded time periods.

The PKI-based key-exposure resilient schemes do not provide support for batch auditing [18]. The computation time of data integrity verification is high and is also linear to the number of data blocks challenged to the cloud server. These schemes also require checking the validity of public key certificates of cloud clients incurring extra computation and communication costs to the verifier. Therefore, to eliminate certificate verification, Wang et al. proposed the Identity-based cloud storage auditing scheme [19] where a third party called the Private Key Generator (PKG) generates the client's auditing secret key from the master secret key of the PKG and the identity of the client. The identity (ID) can be the e-mail address or the IP address of the client. Wang also proposed an Identity-based auditing scheme for multicloud storage [20]. The Identity-based scheme with perfect data privacy [21] realizes zero-knowledge privacy against the Third Party Auditor (TPA). The Identity-based proxy oriented cloud auditing scheme [22] achieves private verification, delegated verification and also public verification. The Identity-based public auditing scheme in [23] provides efficient batch auditing. The Identity-based auditing scheme of Zhang et al. [24] for shared data provides efficient user revocation. In this scheme, the manager and the members of the group share the public key and the auditing secret key. The manager and the non-revoked group users update the secret key whenever a group member gets revoked. The Fuzzy Identity-based scheme [25] uses the biometric data of

the client as the client's identity for improved security. This biometric-based identity increases the computational cost of the cloud client, the cloud server, and the TPA. Many of these Identity-based schemes do not consider the auditing secret key-exposure. The Identity-based scheme in [26] is a lattice-based cloud auditing scheme and provides only forward security. It shows that the auditing secret key exposure is a critical issue in cloud storage auditing and needs to be handled efficiently.

1.3. Objectives of the Work. In the study of the current research on the key exposure issue of cloud storage auditing, it is observed that the following aspects need to be focused on while proposing any mechanism to handle data integrity with acceptable performance.

(1) Generally, trust is a critical issue with Cloud Service Provider (CSP). Auditing secret key exposure is a critical problem and it exists in Identity-based cloud auditing systems. It is practically difficult to prevent key exposure. Existing Identity-based solution provides only forward security and must be improved further to control the impact of auditing secret key exposure. Moreover, the solution for the critical problem should not largely affect the computation time of verifying the data blocks.

(2) Auditing is a complex task for any client. Generally, the client delegates the auditing process to the Third Party Auditor (TPA). In this case, it is necessary to ascertain that the content of the stored data should not be known to the third party.

(3) The TPA can have auditing jobs from multiple clients. Auditing jobs one by one is time consuming since it involves considerable amount of computation. So it is indispensable to use suitable mechanism to improve the efficiency of auditing.

Hence, the primary objective of this work is to propose a PDP-based auditing scheme to protect the security of Identity-based cloud storage auditing in time periods both earlier and later to the time period of the exposure of auditing secret key and also to provide support for batch auditing.

1.4. Contributions. This research focuses on the above aspects of auditing in the public cloud. The contributions of this work are listed as follows:

(1) An Identity-based cloud auditing scheme is proposed using bilinear pairings. The scheme is based on the PDP model. The scheme provides both forward and backward security. Hence, it is strongly resilient to auditing secret key exposure, and the security of cloud storage auditing is protected both earlier and later to the time period of the exposed key. The lifetime of a data file, to be stored in the cloud, is divided into T time periods, and the auditing secret key is jointly updated in each time period by both the client and another entity at the client's end. Key update time is constant, and the number of time periods is unbounded so that the execution time of the algorithms is independent of T. The public keys remain fixed. The scheme does blockless verification with significant reduction in the computation time.

(2) The intractability of solving the discrete logarithm problem and the frequent monitoring of audit reports by the client help to ensure that the TPA acquires nothing from the stored file.

(3) The scheme is extended to include batch auditing. It enables the TPA to audit data files of multiple clients simultaneously, thereby improving the efficiency of auditing further.

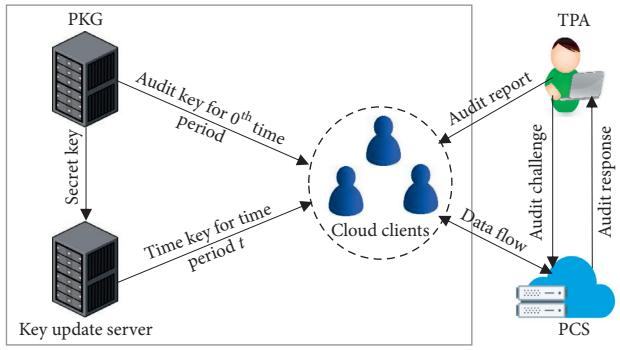
The rest of the paper is structured as follows: Section 2 discusses the system model, the security model, and the preliminaries. Section 3 explains the proposed scheme in detail. In Sections 4 and 5, we demonstrate the security analysis and the performance analysis of the proposed scheme followed by the conclusion in Section 6.

2. Models and Definitions

This section provides the system model, the definition of the strong key-exposure resilient cloud auditing scheme, the security model, and the preliminaries which are used in the system.

2.1. System Model. A basic Identity-based cloud auditing model comprises of the cloud server, one or more cloud clients, the PKG, and the TPA. The cloud server has huge storage capability and stores the data of cloud clients. The PKG is usually a third party. The PKG generates the system parameters, the master secret key, and the system public key. The PKG computes the auditing secret key of the client from the "ID" of the client. The cloud client divides the data file into data blocks of fixed size and generates an authentication tag for each data block using the auditing secret key obtained from the PKG. The client then uploads the data blocks and the tags to the cloud and then deletes them from the local machine. During audit, the TPA sends an audit challenge to the cloud server and verifies the audit response. Audit reports are sent to the clients for review.

Figure 1 illustrates the proposed System Model. In this model, the cloud server is considered to be a Public Cloud Server (PCS). The PKG is presumed to be a server in the IT department of the enterprise. The cloud clients are assumed to be the employees of the enterprise who are entrusted to store the corporate data in the PCS. In the proposed design, all cloud clients have two secret keys, the Audit key and the Time key. The auditing secret key is called the Audit key, "$\mathrm{sk}_{u,t}$." It is different for each client. The additional secret key, the Time key "$\mathrm{sk}_{h,t}$" is used to aid in strong key-exposure resilience. It is also different for each client. It may be assumed that each cloud client is responsible for a single file. Each file F is divided into n number of fixed-size data blocks, $F = \{m_1, m_2, \ldots, m_i, \ldots, m_n\}$ where i is the index of a block. The lifespan of the file is divided into discrete time periods 0, 1, 2, ... The two secret keys are updated in each time period. The public keys remain fixed. The PKG computes the Audit key of all the clients only for time period "0". In



PKG – Private key generator
TPA – Third party auditor
PCS – Public cloud server

Figure 1: The proposed system model.

addition to the entities mentioned above, the proposed model encompasses one more entity called the Key update server. The purpose of this server is to generate the Time key of all the clients for all the time periods. The Audit key of a client of a particular time period is updated by the client using the Time key of the corresponding time period. To authenticate the data blocks of a time period, the client uses only the Audit key of that particular period. The TPA in the proposed model performs both single-client auditing as well as batch auditing.

2.2. Definitions and Security Model

Definition 1 (Strong key-exposure resilient cloud auditing protocol). The proposed strong key-exposure resilient cloud auditing scheme consists of the following algorithms:

(i) Setup $(1^k) \longrightarrow (params, x, y)$: This algorithm is run by the PKG. The algorithm takes the security parameter k and sets up the system parameters. It sets the secret key x and the public key pk_s of the PKG called the master secret key and the system public key, respectively. Also computes the key update server's secret key y and the key update server's public key pk_h. Both pk_s and pk_h are included in the public parameter, *params*.

(ii) InitialAuditKeyGen $(params, x, y, ID) \longrightarrow (sk_{u,0})$: The algorithm generates $sk_{u,0}$, which is the Audit key for the initial time period from the identity ID of the cloud client using *params*, x and y. It is run by the PKG.

(iii) TimeKeyUpdate $(params, y, ID, t) \longrightarrow (sk_{h,t})$: At the beginning of time period t, this algorithm updates the Time key $sk_{h,t}$ of a cloud client with identity ID and secret key y and sends it to the client over a secure channel. It is run by the key update server.

(vi) AuditKeyUpdate $(params, ID, t, sk_{h,t}, sk_{u,t-1}) \longrightarrow (sk_{u,t})$: This algorithm is run by the client. It generates the Audit key $sk_{u,t}$ for the current time period t from the Time key of the current time period $sk_{h,t}$ and the Audit key of the previous time period $sk_{u,t-1}$ and then deletes the keys $sk_{h,t}$ and $sk_{u,t-1}$.

(v) TagGen $(params, F, ID, t, sk_{u,t}) \longrightarrow (F, \sigma_1, \ldots, \sigma_n, t, R)$: This algorithm is run by the cloud client. It generates the authentication tags for a set of data blocks of file F in time period t using the audit key $sk_{u,t}$ of time period t. For each data block m_i of file F, the authentication tag is a pair (σ_i, t). The Audit key of time period "0" is $sk_{u,0}$. It is computed by the PKG during system setup. It is generally not used to authenticate data blocks. *FT* is the file tag of F.

(vi) ChallengeGen $(FT) \longrightarrow (chal, t)$: This algorithm is run by the TPA. It generates the audit challenge $(chal, t)$ which is a random subset of the data block indices of file F and the time period t of the data blocks.

(vii) ProofGen $((chal, t), F, \Phi) \longrightarrow (P)$: This algorithm is run by the PCS. For the given audit challenge $(chal, t)$, the algorithm generates the audit response P from file F of a client and its associated set of authenticators Φ.

(viii) ProofVerify $(params, (chal, t), P, ID) \longrightarrow (0/1)$: The TPA runs this algorithm and verifies the validity of the audit proof P for time period t. The algorithm returns 0 if integrity of the stored blocks is compromised; otherwise, it returns 1.

The following two algorithms, BatchProofGen and BatchProofVerify, are used only in batch auditing.

(xi) BatchProofGen $(\{(chal_k, t_k)\}, \{F_k\}, \{\Phi_k\}) \longrightarrow (P)$: Like ProofGen, this algorithm is run by the PCS. $\{F_k\}$ is the set of files of all the cloud clients. For each client k, the algorithm generates the audit proof corresponding to the audit challenge $(chal_k, t_k)$, from file F_k and its respective authenticator set Φ_k. Then the audit proofs of all the cloud clients are aggregated into a single audit response P and sent to the TPA.

(x) BatchProofVerify $(params, \{(chal_k, t_k)\}, P) \longrightarrow (0/1)$: This algorithm is run by the TPA to verify the audit response P generated by the BatchProofGen algorithm. Like ProofVerify, the algorithm returns 1 if the verification passes and 0 if the verification fails.

In the security model, the Private Key Generator (PKG) is considered to be a reliable body and the Third Party Auditor (TPA) is presumed to be a true and inquisitive entity. The TPA executes the auditing jobs sincerely and is also curious to know about the data. The cloud clients are considered to be partially trusted, i.e., some of the cloud clients may try to forge the authenticators of others. The Cloud Service Provider (CSP) is assumed to be a semitrusted party, i.e., the CSP does not have intentions to delete or alter the data, but some inevitable data loss incidents may trigger to do so in order not to lose fame and popularity. The proposed cloud auditing scheme has the following security properties:

(i) Correctness: If all the authentication tags in the audit response (corresponding to the audit challenge) are valid, then the audit response will always pass the verification.

(ii) Strong key-exposure resilience: The Time key and the Audit key are updated by two different entities, the Time key by the key update server and the Audit key by the client. The updation of the Audit key of a particular time period requires the Time key of the same time period and the Audit key of the previous time period. Hence, even if the Audit key of a particular time period is exposed, the adversary will not be able to obtain any of the earlier or future Audit keys without the corresponding Time key. Therefore, the scheme is strongly resilient to key exposure, and the adversary will not be able to forge the data block tags that are generated earlier or later to the time period of the exposed key.

(iii) Resistance to replace attack: The CSP may try the replace attack, where the PCS swaps a corrupted data block and its tag (m_j, σ_j) in the proof with a valid data block and tag (m_l, σ_l) and try to cheat the auditor. Such a proof will fail during verification due to the usage of collision-resistant hash function.

(vi) Privacy preserving: Audit of cloud data does not reveal the content of stored data to the TPA. The hardness assumption of discrete logarithm problem and the insertion of random coefficients in the audit challenge/response help to achieve this property.

In order to validate the security for strong key-exposure resilience, a Tag-Forge game is defined as follows:

Tag-Forge game: The game is played by two entities, the adversary A and the challenger C. The game consists of three phases, the Setup phase, the Query phase, and the Forgery phase.

Setup phase: Here, the challenger takes the security parameter "k" and sets the master secret key x, the key update server's secret key y, and the public parameters "params." It sends params to "A." The secret keys are kept confidential.

Query phase: Once the system setup is over, A enters into the next phase called the Query phase where A queries C and C answers A. In this phase, A is allowed to query the H_1 and the H_2 oracles. A can also query the Audit key of any cloud client—the Audit key of initial time period and also the Audit keys of other time periods. A can also query the authentication tag of any client's data block.

For any client i, A can query the following oracles with respective inputs:

(i) The H_1 and the *InitialAuditKey* oracles with the input ID_i

(ii) The H_2 and the Auditkey oracles with the input (ID_i, t_i)

(iii) The *TagGen* oracle with the input $(\mathrm{ID}_i, t_i, m_{il})$

A is not allowed to query the Time Key of any client for any time period. A is also not allowed to query the Audit Key of time period t^* of a client of identity ID^*. For the client with identity ID^*, the computation of the Audit key of time period t^* requires the Time key of time period t^* and the Audit key of previous time period $(t^* - 1)$. Even if the adversary has the Audit key of $(\mathrm{ID}^*, t^* - 1)$, the adversary will not be able to compute the Audit key of (ID^*, t^*) without the Time key of (ID^*, t^*). So the adversary will not be able to generate valid authentication tags for the data blocks of time period t^* which is later than time period $t^* - 1$ (backward security).

Forgery phase: After A is finished with its intended set of queries, the Forgery phase is entered. Here, C challenges A with a data block m^* of a client of identity ID^* and time period t^* such that

(1) (ID^*, t^*) was not used earlier by A in AuditKey queries

(2) $(m^*, \mathrm{ID}^*, t^*)$ was not used by A for TagGen queries

A computes a proof (σ^*, R^*, t^*) and wins the game only if (σ^*, R^*, t^*) is a valid authentication tag on data block m^* for time period t^*.

Definition 2 (Strong key-exposure resilience). A cloud auditing protocol is said to be strongly resilient to key-exposure, only if the probability with which the adversary A (Cloud) wins the Tag-Forge game is negligible.

2.3. Preliminaries

(i) Bilinear Map

The bilinear map is used to verify the correctness of the audit response. Let G_1 and G_2 be two multiplicative cyclic groups of same prime order q. The map e: $G_1 \times G_1 \longrightarrow G_2$ is called a bilinear map [30], if it satisfies the following properties:

(a) Bilinearity: for all $s, t \in G_1$ and $a, b \in Z_q^*$, $e(s^a, t^b) = e(s, t)^{ab}$

(b) Nondegeneracy: there exists $g1, g2 \in G_1$ such that $e(g1, g2) \neq 1$

(c) Computability: for all $s, t \in G_1$ one can compute $e(s, t) \in G_2$ in polynomial time

(ii) CDH problem

Given (g, g^a, g^b), where g is a generator of multiplicative group G_1 with order q, and $a, b \in Z_q^*$, the CDH problem [30] is to compute g^{ab}. The CDH assumption is that it is computationally intractable to compute g^{ab}. It is used in security analysis, for the implication of strong key-exposure resilience.

3. Description of the Proposed Cloud Auditing Scheme

The proposed strong key-exposure resilient cloud auditing scheme uses the system model and the bilinear map presented above in Section 2. The scheme uses the key update technique of Wu et al.'s general signature scheme [31] and is extended to support blockless verification of random sets of data blocks which is a key feature of cloud auditing. The algorithms of the proposed scheme are described below:

(i) Setup

(1) Given the input parameter k, the PKG selects two multiplicative cyclic groups G_1, G_2 such that G_1 and G_2 are of large prime order q and $q > 2^k$.
(2) The PKG then selects a symmetric bilinear map, e: $G_1 \times G_1 \longrightarrow G_2$.
(3) Define three cryptographic hash functions, H_1, H_2: $\{0,1\}^* \longrightarrow G_1$ and H_3: $\{0,1\}^* \longrightarrow Z_q^*$.
(4) The PKG randomly selects x, $y \in Z_q^*$ and computes $\mathrm{pk}_s = g^x \in G_1$ and $\mathrm{pk}_h = g^y \in G_1$. Here, g and u are two generators of G_1. x is the master secret key of PKG and pk_s is the corresponding system public key. y and pk_h are the secret and public keys of the key update server, respectively.
(5) The PKG then publishes the public *params* (G_1, G_2, e, g, u, H_1, H_2, H_3, pk_s, pk_h).

(ii) InitialAuditKeyGen

(1) Let the initial time period be "0"
(2) Given a client's identity $\mathrm{ID} \in \{0,1\}^*$, the PKG computes $\mathrm{sk}_{u,0}$ which is the Audit key of the client for initial time period as

$$\mathrm{sk}_{u,0} = H_1(\mathrm{ID})^x \cdot H_2(\mathrm{ID} \parallel 0)^y \in G_1. \tag{1}$$

(3) $\mathrm{sk}_{u,0}$ is then sent over a secure channel to the cloud client
(4) The cloud client receives the key and validates it using

$$e\left(g, \mathrm{sk}_{u,0}\right) = e\left(\mathrm{pk}_s, H_1(\mathrm{ID})\right) e\left(\mathrm{pk}_h, H_2(\mathrm{ID} \parallel 0)\right). \tag{2}$$

(iii) TimeKeyUpdate

(1) At the beginning of time period t, the key update server updates the Time key, $\mathrm{sk}_{h,t}$ using

$$\mathrm{sk}_{h,t} = \left[H_2(\mathrm{ID} \parallel t).\left(H_2(\mathrm{ID} \parallel t-1)\right)^{-1}\right]^y \in G_1. \tag{3}$$

(2) This Time key is forwarded over a secure channel to the client
(3) The cloud client checks the validity of the Time key using

$$e\left(g, \mathrm{sk}_{h,t}\right) = e\left(\mathrm{pk}_h, H_2(\mathrm{ID} \parallel t) \cdot \left(H_2(\mathrm{ID} \parallel t-1)\right)^{-1}\right). \tag{4}$$

(iv) AuditKeyUpdate

(1) The cloud client computes the Audit key $\mathrm{sk}_{u,t}$ for time period t using the time key of the current time period $\mathrm{sk}_{h,t}$ and the Audit key of the previous time period $\mathrm{sk}_{u,t-1}$:

$$\mathrm{sk}_{u,t} = \mathrm{sk}_{h,t} \cdot \mathrm{sk}_{u,t-1} \in G_1. \tag{5}$$

(2) The client then deletes both $\mathrm{sk}_{h,t}$ and $\mathrm{sk}_{u,t-1}$

(v) TagGen

(1) The client picks $r \in Z_q^*$ at random and computes $R = g^r \in G_1$
(2) For each data block $m_i \in Z_q^*$ of time period t, the client computes the hash value h_i:

$$h_i = H_3(\mathrm{ID} \| t \| i) \in Z_q^*. \tag{6}$$

(3) The client then computes the data block tag σ_i using the Audit key $\mathrm{sk}_{u,t}$.

$$\sigma_i = \mathrm{sk}_{u,t}^{h_i} \cdot u^{m_i \cdot r} \in G_1. \tag{7}$$

(4) The client then uploads the file F, the file tag $FT = \mathrm{file\_name} \parallel n \parallel t \parallel \mathrm{Sign}_{\mathrm{sk}}(\dot{f})$ and all the authentication tags $\Phi = \{\sigma_1, \ldots, \sigma_n, t, R\}$ to the PCS. Here, Sign is a digital signature on $\dot{f}$ and $\dot{f} = (\mathrm{file\_name} \| n \| t)$. sk and pk are the secret key and the public key associated with Sign [15].
(5) The PCS validates each data block tag as

$$e(g, \sigma_i) = e\left(R^{m_i}, u\right) e\left(\mathrm{pk}_s, H_1(\mathrm{ID})^{h_i}\right) \cdot e\left(\mathrm{pk}_h, H_2(\mathrm{ID} \parallel t)^{h_i}\right). \tag{8}$$

(vi) ChallengeGen

(1) The TPA checks whether the file tag FT is valid by checking the validity of $\mathrm{Sign}_{\mathrm{sk}}(\dot{f})$ using the public key pk.
(2) If it is not valid, the TPA reports to the client.
(3) If it is valid, the TPA selects a number $l \in Z_q^*$ and generates a subset I containing c number of random file block indices of time period t. The TPA then sends the challenge (*chal*, t) to the PCS, where $chal = \{I, l\}$ and t is the time period.

(vii) ProofGen

(1) The PCS on receiving (*chal*, t) computes the coefficient $v_i \in Z_q^*$ for each data block $i \in I$ using $v_i = l^i \bmod q$
(2) It then aggregates the authenticators of all block indices involved in the challenge, $\sigma = \prod_{i \in I} \sigma_i^{v_i} \in G_1$
(3) And it computes $R^1 = R^{\Sigma_{i \in I}(m_i v_i)}$ and sends the proof $P = \{R^1, t, \sigma\}$ to the TPA

(viii) ProofVerify

The algorithm verifies the audit response of time period t of a single user without downloading the data blocks from the cloud.

(1) On receiving the audit response P from the PCS, the TPA computes $h_i = H_3(\mathrm{ID} \| t \| i)$ for $i \in I$
(2) The TPA then verifies whether

$$e(g, \sigma) = e\left(R^1, u\right) e\left(\mathrm{pk}_s, H_1(\mathrm{ID})^{\Sigma_{i \in I}(h_i v_i)}\right) \cdot e\left(\mathrm{pk}_h, H_2(\mathrm{ID} \parallel t)^{\Sigma_{i \in I}(h_i v_i)}\right). \tag{9}$$

If verified, returns true and false otherwise. The proof of correctness is presented below.

Proof of correctness:

$$
\begin{aligned}
e(g,\sigma) &= e\left(g, \prod_{i\in I} \sigma_i^{v_i}\right)\\
&= e\left(g, \prod_{i\in I} \left(\mathrm{sk}_{u,t}^{h_i} \cdot u^{m_i \cdot r}\right)^{v_i}\right)\\
&= e\left(g, \prod_{i\in I} u^{m_i v_i \cdot r}\right) e\left(g, \prod_{i\in I} \mathrm{sk}_{u,t}^{h_i v_i}\right)\\
&= e\left(g^r, \prod_{i\in I} u^{m_i v_i}\right) e\left(g, \prod_{i\in I} \left(\mathrm{sk}_{h,t} \cdot \mathrm{sk}_{u,t-1}\right)^{h_i v_i}\right)\\
&= e\left(R, u^{\Sigma_{i\in I}(m_i v_i)}\right) e\left(g, \prod_{i\in I} \left(\left[H_2(\mathrm{ID} \parallel t) \cdot \left(H_2(\mathrm{ID} \parallel t-1)\right)^{-1}\right]^y \cdot \mathrm{sk}_{h,t-1} \cdot \mathrm{sk}_{u,t-2}\right)^{h_i v_i}\right)\\
&= e\left(R^{\Sigma_{i\in I}(m_i v_i)}, u\right) e\left(g, \prod_{i\in I} \left(\left[H_2(\mathrm{ID} \parallel t) \cdot \left(H_2(\mathrm{ID} \parallel t-1)\right)^{-1}\right]^y \left[H_2(\mathrm{ID} \parallel t-1) \cdot \left(H_2(\mathrm{ID} \parallel t-2)\right)^{-1}\right]^y \cdot \mathrm{sk}_{h,t-2} \cdot \mathrm{sk}_{u,t-3}\right)^{h_i v_i}\right)\\
&= e\left(R^1, u\right) e\left(g, \prod_{i\in I} \left(\left[H_2(\mathrm{ID} \parallel t) \cdot \left(H_2(\mathrm{ID} \parallel t-2)\right)^{-1}\right]^y \cdot \mathrm{sk}_{h,t-2} \cdot \mathrm{sk}_{u,t-3}\right)^{h_i v_i}\right)\\
&= e\left(R^1, u\right) e\left(g, \prod_{i\in I} \left(H_2(\mathrm{ID} \parallel t)^y \cdot H_1(\mathrm{ID})^x\right)^{h_i v_i}\right)\\
&= e\left(R^1, u\right) e\left(g^x, \prod_{i\in I} H_1(\mathrm{ID})^{h_i v_i}\right) e\left(g^y, \prod_{i\in I} H_2(\mathrm{ID} \parallel t)^{h_i v_i}\right)\\
&= e\left(R^1, u\right) e\left(\mathrm{pk}_s, H_1(\mathrm{ID})^{\Sigma_{i\in I}(h_i v_i)}\right) e\left(\mathrm{pk}_h, H_2(\mathrm{ID} \parallel t)^{\Sigma_{i\in I}(h_i v_i)}\right).
\end{aligned}
\tag{10}
$$

Since the above equation holds, the valid proof can be accepted.

Batch auditing is incorporated in the proposal by modifying the above ProofGen and ProofVerify algorithms to support multiple clients. These algorithms after modification are called BatchProofGen and BatchProofVerify, respectively. In these algorithms, let K be the set of cloud clients of firm "A" and client $k \in K$. ID_k is considered to be the identity of cloud client k. For each client k, the algorithms InitialAuditKeyGen, TimeKeyUpdate, AuditKeyUpdate, TagGen, and ChallengeGen are the same as given above.

(i) BatchProofGen $(\{(chal_k, t_k)\}, \{F_k\}, \{\Phi_k\}) \longrightarrow (P)$

(1) The PCS computes $\sigma^* = \amalg_{k\in K} \amalg_{i\in I} \sigma_{ki}^{v_{ki}} \in G_1$ from $\{\Phi_k\}$

(2) Then the PCS computes $R^* = \amalg_{k\in K} R_k^{\Sigma_{i\in I}(m_{ki} v_{ki})} \in G_1$ using $\{F_k\}$ and $\{\Phi_k\}$

(3) The batch proof/response, $P = \{R^*, \{t_k\}_{k\in K}, \sigma^*\}$ is sent to the TPA

(ii) BatchProofVerify $(params, \{(chal_k, t_k)\}, P) \longrightarrow (0/1)$

Similar to ProofVerify, BatchProofVerify verifies the data blocks without downloading them.

(1) For each cloud client k included in the batch, the TPA computes the hash value $h_{ki} = H_3(\mathrm{ID}_k \| t_k \| i) \in Z_q^*$ for each data block i, indexed in the challenge $chal_k$.

(2) The TPA then verifies the files of all clients using a single equation and returns 1 if L.H.S = R.H.S., otherwise returns 0.

$$
\begin{aligned}
e(g,\sigma^*) = e\left(R^*, u\right) e\left(\mathrm{pk}_s, \prod_{k\in K} H_1(\mathrm{ID}_k)^{\Sigma_{i\in I}(h_{ki} v_{ki})}\right)\\
\cdot e\left(\mathrm{pk}_h, \prod_{k\in K} H_2(\mathrm{ID}_k \parallel t)^{\Sigma_{i\in I}(h_{ki} v_{ki})}\right).
\end{aligned}
\tag{11}
$$

4. Security Analysis

The security proofs for the audit response in batch auditing are similar to the audit response in a single client setting. Hence, in this section, the audit response in a single client setting is considered for analyzing the algorithms of the

proposed scheme. Here, some theorems are formalized and proofs for the same are presented.

Theorem 1 (Strong key exposure resilience). *If the CDH assumption holds in bilinear group G_1, then the proposed Identity-based cloud auditing scheme is strongly resilient to key exposure.*

Proof. Consider G_1 to be a multiplicative group of prime order q. Let g be a generator of G_1 and $a, b \in Z_q^*$. The CDH problem is to compute the value of g^{ab}, given the values of g, g^a, and g^b. The proof for the theorem is established using the Tag-Forge game explained in Section 2. The game is played by two entities, A and C. Here, A is an adversary and C is the challenger. If A wins the game with nonnegligible probability ϵ, then C can solve the CDH problem.

In this game, H_1 and H_2 are considered to be random oracles and H_3 is considered as a one-way function with collision-resistance. C maintains the lists H_1, H_2, L_1, L_2, and T, which are initially empty. For any query on InitialAuditKey or AuditKey or TagGen oracle on identity ID_i of a cloud client for time period t, a H_1 query is assumed to be already been made on that identity.

(i) Setup: Let G_1 and G_2 be two cyclic multiplicative groups of prime order q. C sets $pk_s = g^a$ as the public key of PKG and $pk_h = g^y$ as the public key of key update server. C randomly chooses $t \in Z_q^*$ and sets $u = g^t$ (t is different from time periods t_i and t^*). C returns (G_1, G_2, e, g, u, q, pk_s, pk_h) to A.

(ii) H_1 Query: The adversary A makes a H_1 query for identity ID_i. If ID_i is already in the H_1 list as a tuple (ID_i, c, h_{i1}, m) (m is different from data blocks m_{il} and m^*), then C returns h_{i1}. If not, C tosses a coin $c \in \{0, 1\}$ with probability $P[c = 0] = \delta$ and $P[c = 1] = 1 - \delta$ and does one of the following:

(1) If $c = 0$, B randomly picks $m \in Z_q^*$ to set $H_1(ID_i) = g^m = h_{i1} \in G_1$
(2) If $c = 1$, B randomly selects $m \in Z_q^*$ and sets $H_1(ID_i) = (g^b)^m = h_{i1} \in G_1$

In both the cases, C adds (ID_i, c, h_{i1}, m) to the H_1 list and returns h_{i1} to A.

(iii) H_2 Query: When A queries H_2 list with (ID_i, t_i) where t_i is time period, C returns the corresponding value if the H_2 list contains an entry for (ID_i, t_i); otherwise, C selects an integer $n \in Z_q^*$ randomly, computes $H_2(ID_i \| t_i) = g^n \in G_1$, and adds ($ID_i$, t_i, n, g^n) to the list and also returns g^n to A.

(iv) InitialAuditKeyQuery: When A queries C with input ID_i, C returns the value $sk_{u,0}$ if a tuple matching the input ID_i exists in the L_1 list. If the ID_i is not found in the list, then C retrieves the tuple (ID_i, c, h_{i1}, m) from the H_1 list and responds as follows:

(1) If $c = 0$, C computes $sk_{u,0} = pk_s^m = (g^a)^m \in G_1$, adds ($ID_i$, $sk_{u,0}$) to the list L_1 and returns $sk_{u,0}$ to A.
(2) If $c = 1$, C halts with an unsuccessful message.

(v) AuditKeyQuery: For the input (ID_i, t_i), C retrieves the tuple (ID_i, t_i, $sk_{u,t}$) from the L_2 list and returns $sk_{u,t}$ to A. If such a tuple does not exist, then the tuple (ID_i, c, h_{i1}, m) is retrieved from the H_1 list.

(1) If $c = 0$, C retrieves the tuple (ID_i, t_i, n, g^n) from the H_2 list, computes $sk_{u,t} = (g^n)^y \cdot (g^a)^m \in G_1$ to A, adds (ID_i, t_i, $sk_{u,t}$) to the list L_1, and returns $sk_{u,t}$ to A.
(2) If $c = 1$, C reports failure and aborts.

(vi) TagGen Query: For A's input (ID_i, t_i, m_{il}), C finds the tuple (ID_i, t_i, m_{il}, σ_{il}, R) in the T list and returns (σ_{il}, R, t_i). If the tuple is not found in the list, then C does any one of the following depending on the value of c of the H_1 tuple (ID_i, c, h_{i1}, m).

(1) If $c = 1$, C outputs a failure message and aborts.
(2) If $c = 0$, C proceeds to compute the tag for block m_{il}. C randomly selects $k \in Z_q^*$ to compute $R = g^k$ and $h_{i3} = H_3(ID_i \| t_i \| l) \in Z_q^*$. Then it computes $\sigma_{il} = ((g^n)^y \cdot (g^a)^m)^{h_{i3}} \cdot u^{m_{il} \cdot k} \in G_1$. The output ($\sigma_{il}$, R, t_i) is a valid authentication tag for data block m_{il}. It is added to the T list and is also returned to A.

(vii) Forgery/output: Finally, A outputs a valid authentication tag (σ^*, R^*, t^*) on the challenged data block m^* (with index i^*) with non-negligent probability ϵ. Since (σ^*, R^*, t^*) is valid, C checks the value of c^* of the H_1 tuple $(ID_i^*, c^*, h_{i1}^*, m^*)$.

(1) If $c^* = 0$, C reports failure and aborts.
(2) If $c^* = 1$, C proceeds to find the value of g^{ab}

$$e(g, \sigma^*) = e\left((R^*)^{m^*}, u\right) e\left(pk_s, H_1(ID^*)^{h^*}\right) \cdot e\left(pk_h, H_2(ID^* \| t^*)^{h^*}\right), \tag{12}$$

where $h^* = (ID^* \| t^* \| i^*), H_1(ID^*) = (g^b)^m$.

$H_2(ID^* \| t^*) = g^n$, and $u = g^t$. Therefore, $e(g, \sigma^*) = e(g, (R^*)^{m^* \cdot t}) e(g^a, (g^b)^{m \cdot h^*}) e(g^y, g^{n \cdot h^*})$, $\sigma^* = (R^*)^{m^* \cdot t} \cdot (g^{ab})^{m \cdot h^*} \cdot g^{y \cdot n \cdot h^*}$, $g^{ab} = [\sigma^* \cdot ((R^*)^{m^* \cdot t} \cdot (pk_h)^{n \cdot h^*})^{-1}]^{(m \cdot h^*)^{-1}}$.

Let us assume that A makes Q_e, Q_c, and Q_t queries, which are InitialAuditKey, AuditKey, and TagGen queries, respectively. The success probability of C depends on the following five events:

$Event_1$: C does not halt due to any of A's InitialAuditKey Queries

$Event_2$: C does not halt due to any of A's AuditKey Queries

$Event_3$: C does not halt due to any of A's TagGen Queries

Event_4: A produces a valid tag

Event_5: Event_4, $c^* = 1$ for the H_1 tuple $(\text{ID}_i^*, c^*, h_{i1}^*, m^*)$

$$\begin{aligned} P[\text{Event}_1 \wedge \text{Event}_2 \wedge \text{Event}_3 \wedge \text{Event}_4 \wedge \text{Event}_5] &= \delta^{Q_e} \cdot \delta^{Q_c} \cdot \delta^{Q_t} \cdot \varepsilon \cdot (1-\delta) \\ &= \delta^{Q_e+Q_c+Q_t} \cdot \varepsilon \cdot (1-\delta) \\ &= \left[\frac{(Q_e+Q_c+Q_t)^{Q_e+Q_c+Q_t}}{(Q_e+Q_c+Q_t+1)^{Q_e+Q_c+Q_t+1}}\right] \cdot \varepsilon, \end{aligned} \tag{13}$$

when $\delta = \dfrac{(Q_e+Q_c+Q_t)}{(Q_e+Q_c+Q_t+1)}$.

Since the above is a contradiction to the CDH assumption, it is impossible to forge the authentication tag. Hence, it is concluded that the proposed Identity-based scheme is a strong key exposure resilient cloud auditing scheme. □

Theorem 2 (Resistance to replace attack). *The proposed cloud auditing scheme can resist the replace attack of the PCS.*

Proof. Let's say that a challenged data block m_j and/or its authentication tag σ_j is corrupted. The PCS uses a valid (m_l, σ_l) instead of (m_j, σ_j) in the proof since (m_j, σ_j) will not pass the auditing. The PCS sends the proof $P = \{R^* = R^{m_l v_j + \Sigma_{i\neq l, i\in j}(m_i v_i)}, t, \sigma^* = \sigma_l^{v_j} \cdot \prod_{i\in I, i\neq j} \sigma_i^{v_i}\}$ to the TPA.

$$\begin{aligned} e(g,\sigma^*) &= e\left(g, \sigma_l^{v_j} \cdot \prod_{i\in I, i\neq j} \sigma_i^{v_i}\right) \\ &= e\left(g, \left(\text{sk}_{u,t}^{h_l} \cdot u^{m_l \cdot r}\right)^{v_j} \cdot \prod_{i\in I, i\neq j} \left(\text{sk}_{u,t}^{h_i} \cdot u^{m_i \cdot r}\right)^{v_i}\right) \\ &= e\left(g, u^{m_l v_j \cdot r} \cdot \prod_{i\in I, i\neq j} u^{m_i v_i \cdot r}\right) e\left(g, \text{sk}_{u,t}^{h_l v_j} \cdot \prod_{i\in I, i\neq j} \text{sk}_{u,t}^{h_i v_i}\right) \\ &= e\left(g^r, u^{m_l v_j} \cdot \prod_{i\in I, i\neq j} u^{m_i v_i}\right) e\left(g, \left(H_2(\text{ID} \parallel t)^y \cdot H_1(\text{ID})^x\right)^{h_l v_j} \cdot \prod_{i\in I, i\neq j} \left(H_2(\text{ID} \parallel t)^y \cdot H_1(\text{ID})^x\right)^{h_i v_i}\right) \\ &= e\left(R, u^{m_l v_j + \Sigma_{i\in I, i\neq j}(m_i v_i)}\right) e\left(g, \left(H_1(\text{ID})^{h_l v_j} \cdot \prod_{i\in I, i\neq j} H_1(\text{ID})^{h_i v_i}\right)^x\right) e\left(g, \left(H_2(\text{ID} \parallel t)^{h_l v_j} \cdot \prod_{i\in I, i\neq j} H_2(\text{ID} \parallel t)^{h_i v_i}\right)^y\right) \\ &= e\left(R^{m_l v_j + \Sigma_{i\in I, i\neq j}(m_i v_i)}, u\right) e\left(g^x, H_1(\text{ID})^{h_l v_j + \Sigma_{i\in I, i\neq j}(h_i v_i)}\right) e\left(g^y, H_2(\text{ID} \parallel t)^{h_l v_j + \Sigma_{i\in I, i\neq j}(h_i v_i)}\right) \\ &= e(R^*, u) e\left(\text{pk}_s, H_1(\text{ID})^{\Sigma_{i\in I}(h_i v_i) + (h_l - h_j) v_j}\right) e\left(\text{pk}_h, H_2(\text{ID} \parallel t)^{\Sigma_{i\in I}(h_i v_i) + (h_l - h_j) v_j}\right). \end{aligned} \tag{14}$$

The above verification will pass only if $h_l - h_j = 0$. Since H_3 is a collision-resistant hash function $h_l - h_j \neq 0$, the verification will fail. □

Theorem 3 (Privacy preserving). *The curious TPA will not be able to get the data stored in the PCS or any information about the data.*

Proof. The TPA receives the proof message, $P = \{R^1, t, \sigma\}$ where $R^1 = R^{\Sigma_{i\in I}(m_i v_i)}$ and $\sigma = \prod_{i\in I} \sigma_i^{v_i}$, from the PCS. Let us consider v_i to be a variable and m_i to be a constant in the linear equation $\Sigma_{i\in I}(m_i v_i)$. Even though the TPA knows the value of all the variables, he will not be able to compute the value of one or more constants since there are more than 100 variables in a single linear equation. The client also monitors the audit report and ensures that subsequent challenges do not contain the same set of block indices. Moreover, even if the TPA comes to know about the value of R^{m_i}, it is impossible to compute m_i due to the intractability of solving the discrete logarithm problem. So the TPA will not be able to gain any information about the cloud data from the proof message.

From the above theorems, it is shown that the proposed cloud auditing scheme is an Identity-based strong key exposure resilient auditing scheme and it is resistant to the replace attack of the PCS. Also, the TPA will not be able to collect any information from the data file during audit. □

5. Performance Analysis

The proposed scheme is compared with three existing cloud storage auditing schemes, namely, Yu et al.'s scheme of [17], Yu et al.'s scheme of [15], and Zhang et al.'s scheme of [24] (called Scheme (2016), Scheme (2017), and Scheme (2018),

Table 1: Comparison of storage size and size of audit messages.

Schemes	Data storage on the cloud server (bytes)	Audit challenge (bytes)	Audit response (bytes)
Scheme (2016)	$\|F\| + (n+2)\|G_1\| + \|t\|$	$c(\|Z_q\| + \|i\|) + \|t\|$	$3\|G_1\| + \|Z_q\| + \|t\|$
Scheme (2017)	$\|F\| + (n+1)\|G_1\| + \|t\|$	$c(\|Z_q\| + \|i\|) + \|t\|$	$2\|G_1\| + \|Z_q\| + \|t\|$
Scheme (2018)	$\|F\| + n\|G_1\| + \|\mathrm{RN}\|$	$c(\|Z_q\| + \|i\|)$	$\|G_1\| + \|Z_q\|$
Proposed scheme	$\|F\| + (n+1)\|G_1\| + \|t\|$	$\|Z_q\| + c \cdot \|i\| + \|t\|$	$2\|G_1\| + \|t\|$

Table 2: Comparison of computation time in terms of number of operations.

Schemes	Updation of secret keys	Tag generation	Proof generation	Proof verification
Scheme (2016)	$T_H + 4T_e + 2T_m$	$n(T_H + 2T_e + 2T_m) + T_e$	$cT_e + cT_m$	$(c+1)T_H + 3T_p + (c+2+bl)T_e + 2T_m$
Scheme (2017)	$2T_H + 2T_e + T_m$	$n(T_H + 2T_e + 2T_m) + T_e$	$cT_e + cT_m$	$(c+1)T_H + 3T_p + (c+2)T_e + 2T_m$
Scheme (2018)	T_e	$n(T_H + 2T_e + T_m)$	$cT_e + cT_m$	$cT_H + 2T_p + (c+3)T_e + (c+4)T_m$
Proposed scheme	$2T_H + T_e + T_{inv} + 2T_m$	$n(2T_e + T_m) + T_e$	$(c+1)T_e + cT_m$	$2T_H + 4T_p + 2T_e$

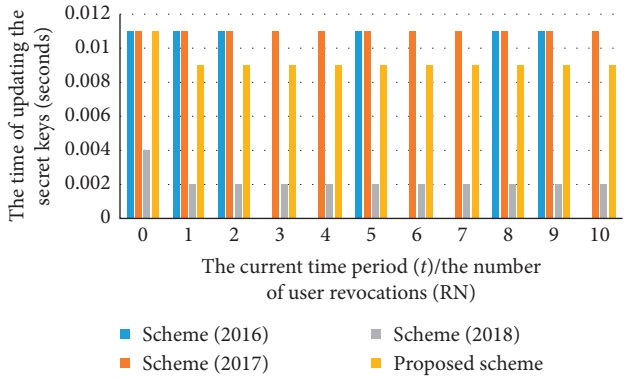


Figure 2: The key update time in different time periods (*for Scheme (2016), the key update time is 0 seconds in time periods 3, 4, 6, 7, and 10).

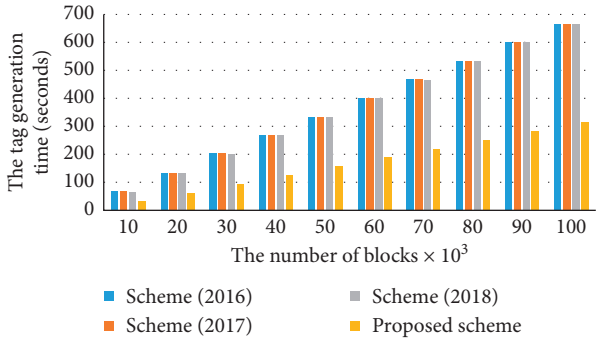


Figure 3: The tag generation time with different number of blocks.

respectively, in Tables 1 and 2 and in Figures 2–5). The schemes in [15, 17] are key-exposure resilient PKI-based schemes. The scheme in [24] is an Identity-based scheme for shared data with efficient user revocation.

5.1. Analysis of the Data Storage Size and the Size of Audit Messages. The numerical analysis of the size of the data stored on the cloud server and the size of audit messages used for communication between the cloud server and the TPA is presented in Table 1 for all the four schemes. The size of the audit response message in batch auditing is given in Table 3 for the proposed scheme. The file and its associated

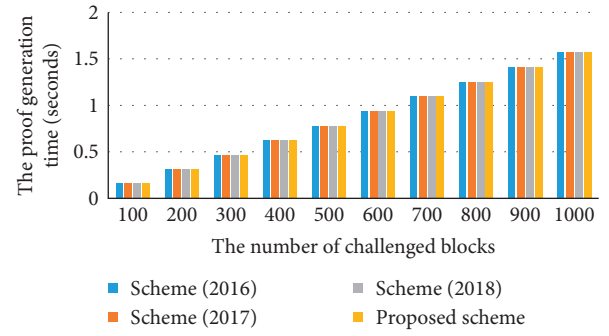


Figure 4: The proof generation time with different number of challenged blocks.

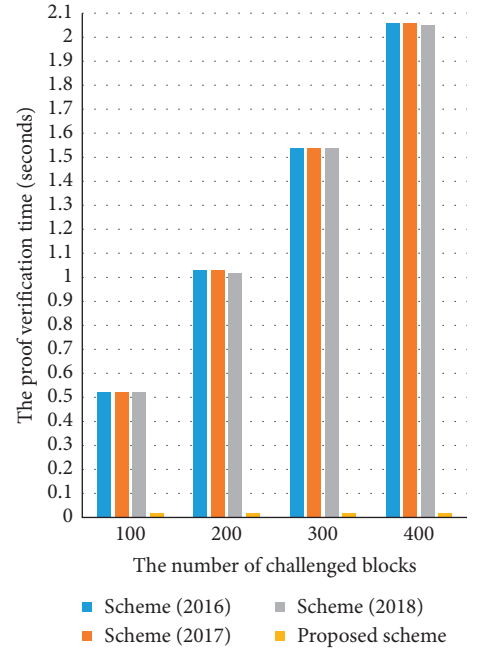


Figure 5: The proof verification time with different number of challenged blocks.

set of authentication tags (one tag per data block of the file) are stored in the cloud server. Therefore, the size of the authentication tags is also included in the measurement of

Table 3: Comparison of size of audit response and computation time in terms of number of operations in batch auditing.

	Proposed scheme
Audit response (bytes)	$2\|G_1\| + \|t\|$. no_of_clients
Proof generation	no_of_clients $[(c+1)T_e + (c+2)T_m]$
Proof verification	no_of_clients $(2T_H + 2T_e) + 4T_p$

the amount of data storage in Table 1. In the tables, $|F|$ denotes the size of a file F, n represents the number of data blocks of F, and $|i|$ is the size of a data block index. $|G_1|$ and $|Z_q|$ denote the size of an element of G_1 and Z_q, respectively. c represents the number of data blocks challenged to the cloud server, $|t|$ is the size of a time period, RN is the number of user revocations, and |RN| represents its size. It is seen from Table 1 that the size of an audit challenge message in the proposed scheme is smaller than the other three schemes.

5.2. Analysis of Computation Time. Table 2 gives the numerical analysis of the computation time of all the four schemes. The computation time of the proposed scheme in batch auditing is given in Table 3. In Table 2, bl is the bit length of a node in the binary tree in [17]. T_H, T_e, T_p, T_{inv}, and T_m denote the execution time of a hash to a point in G_1, exponentiation in group G_1, pairing from G_1 to G_2, inverse in G_1, and multiplication in G_1, respectively. Other operations such as set operations, operations in the group Z_q, are omitted in the tables as they contribute negligible computation time.

All the four schemes are simulated and tested on Ubuntu Linux 17.04 with Intel(R) Core(TM) i5 CPU 650 @ 3.20 GHz and 4.00 GB RAM. For cryptographic operations, Pairing-Based Cryptography (PBC) Library version 0.5.14 [32] and a bilinear map that uses a supersingular elliptic curve are used. In all the schemes, $|G_1| = 128$ bytes and $|Z_q| = 20$ bytes. $|F|$ and n are assumed to be 20 MB and 1,000,000, respectively, as the schemes in [15, 24].

The probability of error detection is determined by the number of data blocks challenged by the TPA. Let us say that s out of n blocks are bad, i.e., corrupted or deleted. The error detection probability is 100%, if all the n blocks are challenged. If c out of n blocks are challenged, then the detection probability is

$$P(c) \geq 1 - \left(1 - \frac{s}{n}\right)^c. \quad (15)$$

With $c = 300$ blocks and $s = 1\%$ of n, P(300) is at least 95%.

The graph in Figure 2 gives the time taken to update the auditing secret key. It is seen that the key update time of Scheme (2018) is much lesser than all the other schemes during some of the user revocations. The purpose of key updation of Scheme (2018) also differs from that of all the other schemes. Scheme (2016), Scheme (2017), and the proposed scheme update the secret key in each time period to achieve resilience to key exposure. Scheme (2018) updates the secret key to revoke a group user. In Scheme (2018), the RN variable is initially set to 0 and the auditing secret key is jointly generated by the PKG, the group manager, and the

Figure 6: The Proof generation time of the proposed scheme in batch auditing for $c = (300 \cdot \text{no\_of\_clients})$ blocks.

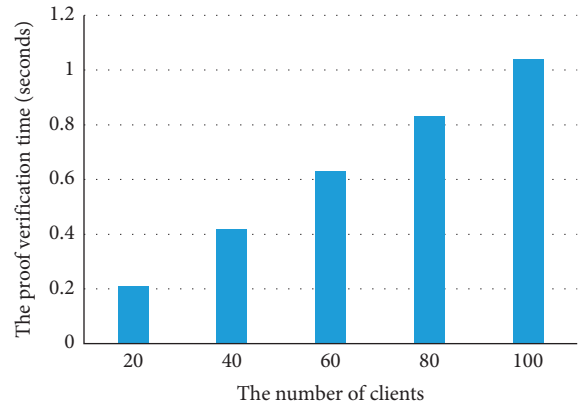


Figure 7: The proof verification time of the proposed scheme in batch auditing for $c = (300 \cdot \text{no\_of\_clients})$ blocks.

group users. During each user revocation, the RN variable gets incremented by 1 and the group manager and the non-revoked group users update the auditing secret key. The key update time is higher for RN = 0 ($2T_e$) than for RN > 0 (T_e). In Scheme (2016), the secret key is updated by the TPA using a binary tree, and therefore, the key update time is 0.011 seconds for the internal nodes of the tree (time periods 0, 1, 2, 5, 8, 9) and 0 seconds for the leaves of the tree (time periods 3, 4, 6, 7, 10). In Scheme (2017), the audit key is jointly updated by both the client and the TPA. In the proposed scheme, it is jointly updated by the key update server and the client. In the proposed scheme, the key update time for the initial time period is $(2T_H + 2T_e + T_m)$ and is higher than the remaining time periods due to the additional T_e. A single T_e is greater than $T_{inv} + 2T_m$.

Figure 3 shows that the time to generate authentication tags in the proposed scheme is lesser than all the other schemes. It is because the tag generation algorithm of the proposed scheme does not have any hash to a point in G_1 operation. It is seen from Figure 4 that the audit response/proof generation time is almost similar for all the schemes. In Figure 5, the Proof verification time of the proposed scheme is constant irrespective of the number of data blocks. It is also very less when compared to the other schemes due to significantly less number of hash to a point in G_1 and exponentiations in G_1. Figures 6 and 7 show that in batch

auditing, both the proof generation time and the Proof verification time of the proposed scheme grow linearly with the number of clients.

6. Conclusion

The need to preserve the integrity of data stored in the cloud rises with the growing demand for storage-as-a-service offering of cloud. So cloud storage auditing schemes are designed to verify the possession of cloud data but there are critical issues in these auditing schemes. This paper studies the auditing secret key exposure problem in Identity-based cloud storage auditing schemes. Since exposure of secret key is undetectable, a better way to handle the key exposure problem is to minimize the damage caused by the exposed key. An Identity-based strong key-exposure resilient cloud auditing scheme using bilinear pairing is designed and implemented. The proposed scheme preserves the security of cloud auditing both before and after the key exposure by forward and backward security mechanism. Batch auditing is also incorporated into the scheme to ease the workload of the auditor. The scheme is provably secure using the computational Diffie–Hellman assumption in the random oracle model. Experimental results show that the proposed scheme is efficient in auditing the data blocks.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors would like to thank Dr.S. Jayaprakash, Emeritus Professor, formerly with IIT Madras, for his constant support and encouragement which helped to complete the work successfully.

References

[1] Prescient & Strategic Intelligence, *Global Enterprise Data Storage*, Prescient & Strategic Intelligence, Noida, India, https://www.psmarketresearch.com/market-analysis/enterprise-data-storage-market.

[2] A. D. Rayome, "69% of enterprises moving business-critical applications to the cloud," 2019, https://www.techrepublic.com/article/69-of-enterprises-moving-business-critical-applications-to-the-cloud/.

[3] T. Singleton, "Why are so many enterprises moving to the cloud?," 2018, https://datometry.com/blog/moving-to-the-cloud-survey-analysis/.

[4] T. Nikl and R. K. Chintalapudi, "8 common reasons why enterprises migrate to the cloud," 2018, https://cloud.google.com/blog/products/storage-data-transfer/8-common-reasons-why-enterprises-migrate-to-the-cloud.

[5] Flexera, "Cloud computing trends," 2019, https://www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2019-state-cloud-survey.

[6] R. Oskoui, "5 Key cloud security challenges," 2018, https://www.cdnetworks.com/cloud-security/5-key-cloud-security-challenges/4208/.

[7] G. Ateniese, R. Burns, R. Curtmola et al., "Provable data possession at untrusted stores," in *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS'07)*, pp. 598–610, Alexandria, VA, USA, November 2007.

[8] A. Juels and B. S. Kaliski, "PORs: proofs of retrievability for large files," in *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS'07)*, pp. 584–597, New York, NY, USA, November 2007.

[9] H. Shachem and B. Waters, "Compact proofs of retrievability," in *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2008)*, pp. 90–107, Melbourne, Australia, December 2008.

[10] H. Wang, "Proxy provable data possession in public clouds," *IEEE Transactions on Services Computing*, vol. 6, no. 4, pp. 551–559, 2013.

[11] Y. Zhu, G. Ahn, H. Hu, and S. Yau, "Dynamic audit services for outsourced storages in clouds," *IEEE Transactions on Services Computing*, vol. 6, no. 2, pp. 227–238, 2013.

[12] B. Wang, B. Li, and H. Li, "Oruta: Privacy-preserving public auditing for shared data in the cloud," in *Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing*, vol. 2, pp. 43–56, Honolulu, HI, USA, June 2014.

[13] B. Wang, H. Li, X. Liu, F. Li, and X. Li, "Efficient public verification on the integrity of multi-owner data in the cloud," *Journal of Communications and Networks*, vol. 16, no. 6, pp. 592–599, 2014.

[14] Y. Zhu, H. Hu, G.-J. Ahn, and M. Yu, "Cooperative provable data possession for integrity verification in multicloud storage," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 12, pp. 2231–2244, 2012.

[15] J. Yu and H. Wang, "Strong key-exposure resilient auditing for secure cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 8, pp. 1931–1940, 2017.

[16] R. Ding, Y. Xu, J. Cui et al., "A public auditing protocol for cloud storage system with intrusion-resilience," *IEEE Systems Journal*, pp. 1–12, 2019.

[17] J. Yu, K. Ren, and C. Wang, "Enabling cloud storage auditing with verifiable outsourcing of key updates," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 6, pp. 1362–1375, 2016.

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

[19] 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. 118–121, 2014.

[20] H. Wang, "Identity-based distributed provable data possession in multicloud storage," *IEEE Transactions on Services Computing*, vol. 8, no. 2, pp. 328–340, 2015.

[21] Y. Yu, M. H. Au, G. Ateniese et al., "Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 4, pp. 767–778, 2016.

[22] H. Wang, D. He, and S. Tang, "Identity-based proxy-oriented data uploading and remote data integrity checking in public cloud," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 6, pp. 1165–1176, 2016.

[23] J. Zhang and Q. Dong, *Efficient ID-based public auditing for the outsourced data in cloud storage*, Vol. 343-344, Elsevier, Information Sciences, Amsterdam, Netherlands, 2016.

[24] Y. Zhang, J. Yu, R. Hao et al., "Enabling efficient user revocation in identity-based cloud storage auditing for shared big data," *IEEE Transactions on Dependable and Secure computing*, pp. 1–13, 2018.

[25] Y. Li, Y. Yu, G. Min, W. Susilo, J. Ni, and K.-K. R. Choo, "Fuzzy identity-based data integrity auditing for reliable cloud storage systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 1, pp. 72–83, 2019.

[26] X. Zhang, H. Wang, and C. Xu, *Identity-Based Key-Exposure Resilient Cloud Storage Public Auditing Scheme From Lattices*, vol. 472, pp. 223–234, Elsevier, Information Sciences, Amsterdam, Netherlands, 2019.

[27] M. Bellare and S. K. Miner, "A forward-secure digital signature scheme," in *Proceedings of the 19th Springer Annual International Conference On Cryptology (Crypto'99), LNCS 1666*, pp. 431–448, Barbara, CA, USA, August 1999.

[28] J. Yu, K. Ren, C. Wang, and V. Varadharajan, "Enabling cloud storage auditing with key-exposure resistance," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 6, pp. 1167–1179, 2015.

[29] B. Wang, B. Li, and H. Li, "Panda: public auditing for shared data with efficient user revocation in the cloud," *IEEE Transactions on Services Computing*, vol. 8, no. 1, pp. 92–106, 2015.

[30] D. Boneh, "A brief look at pairings based cryptography," in *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS '07)*, pp. 19–26, Providence, RI, USA, October 2007.

[31] T. Wu, Y. Tseng, and C. Yu, "ID-based key-insulated signature scheme with batch verifications and its novel application," *International Journal of Innovative Computing, Information and Control*, vol. 8, no. 7, pp. 4797–4810, 2012.

[32] B. Lynn, *The Pairing-Based Cryptographic Library*, https://crypto.stanford.edu/pbc/, 2018.