Public Integrity Verification for Cloud Storage with Efficient Key-update

Hao Yan[1,2,3], Yanan Liu[1], Dandan Huang[1], Shuo Qiu[1], Zheng Zhang[1]*

[1] School of Network Security, Jinling Institute of Technology, China
[2] Xi'an Innovation College of Yan'an University, China
[3] Fujian Provincial Key Laboratory of Network Security and Cryptology, Fujian Normal University, China
pxy_hao@jit.edu.cn, yanan.liu@jit.edu.cn, huangdd@jit.edu.cn, shuoqiu@jit.edu.cn, zhangzheng@jit.edu.cn

Abstract

To improve the security of the data on cloud storage, numbers of data integrity auditing schemes have been proposed in the past several years. However, there only a few schemes considered the security challenge that the user's key is exposed unknowingly which is very likely to happen in real-life. To cope with the problem, we propose a public data integrity auditing scheme for cloud storage with efficient key updating. In our scheme, the user's key is updated periodically to resist the risk of key exposure. Meanwhile, the authentication tags of blocks are updated simultaneously with the key updating so as to guarantee the data integrity can be verified normally. The algorithm of key updating in our scheme is very efficient which only needs a hash operation while previous schemes need two or three exponentiation operations. Moreover, the workload of tag updating is undertaken by cloud servers with a re-tag-key which reduces the burden of users and improves the efficiency of the scheme. The communication cost of the scheme is also reduced greatly, for instance, the information size in 're-key' step is decreased from two group members to one. Furthermore, we give the formal security model of our scheme and prove the security under the CDH assumption. The experimental results show that our proposal is efficient and feasible.

Keywords: Cloud storage, Data integrity checking, Key update

1 Introduction

With the increasing requirement for data storage, cloud storage service has attracted extensive interest and attention from all the fields of industry because of its flexibility, reliability and scalability features [1]. With cloud storage, clients no need to bear expensive cost of building infrastructure and employing staffs, they can easily enjoy the service of storing and maintaining great amount of data with lower investment. Moreover, the ubiquitous Internet makes client to access the data in cloud very conveniently. As a result, more and more clients including enterprises or personal choose to outsource data in cloud server [2-3]. However, because the client loses the physical control of the data and at the same time the cloud service provider (CSP) is untrusted either, the concern about the integrity of the data is increasing that client worries about whether the data is kept intact by CSP [4-5]. Thus, to verify the security of the data on CSP becomes an important and urgent demand for the client [6].

To address this problem, many provable data possession (PDP) schemes have been proposed [6-30]. In these schemes, the whole data is split to many small blocks and each block is signed by client. Then all the blocks and its authenticator tag are uploaded to CSP. Since the block tag is computed based on its block, client can verify the rightness of the data block by auditing the validity of its tag. Obviously, if the block tag cannot pass the audition, it means the block is not correct. To reduce the cost of communication and computation, PDP utilizes the probability checking idea that by auditing the integrity of a set of blocks randomly selected to get the integrity of the whole data. It is proved that PDP model achieves high error detection probability of the data with very low cost. Many researchers have done a lot of work for designing PDP schemes for different application scenario, for instance, some schemes focus on data dynamic, some schemes consider the integrity checking for group shared data and so on. However, there is an important secure problem in data integrity checking has not been given enough attention, that is, if the client's key is exposed, how to ensure the soundness of the check result for data integrity. In fact, the problem of key exposures seems to be unavoidable in the real work due to the complex network environment and social environment. Therefore, the data integrity scheme should take into account the great threat of key-exposure to improve the security and practicability.

1.1 Problem Statement

Resisting the key-exposure is an import security requirement for PDP schemes, because if the key is exposed, the data on cloud storage can be forged, modified even deleted by the untrusted cloud server and other adversaries while the data auditor can not know these bad accidents at all. To realize the key-exposure resistance, the main idea is to update the key periodically that each key is used only for a fix time period after which a new key is generated to replace the old one to continue the next work. Therefore, the threaten of key-exposure is reduced greatly. For a PDP scheme with key-exposure resistant, the key and the block tag should be updated simultaneously, otherwise, the scheme is invalid.

*Corresponding Author: Hao Yan; E-mail: pxy_hao@jit.edu.cn
DOI: 10.53106/160792642023112406009

1.2 Contributions

To address the key-exposure problem, the paper presents a public data integrity checking scheme with efficient key-update. In our scheme, client's private key is updated periodically and the tags of data blocks are also re-generated with the new key. Therefore, even if the adversary gains the exposed key of the client, our scheme still keeps secure and effective. The contributions of the paper are summarized as follows:

(1) We present a public data integrity verification scheme for cloud data which can efficiently audit the data integrity remotely. Moreover, out scheme updates client's private key periodically to resist the key-exposure attack while the block tags are re-generated by CSP to maintain the data integrity checking regardless the times of key updating.

(2) We formalize the system model of the data integrity auditing scheme with key-update. We define the security model of the scheme through a security game and prove that the scheme is able to resist the key exposure attack under the random oracle model.

(3) We give the performance analysis of our scheme, and the experiment results demonstrate that our scheme can efficiently audit the integrity of data on cloud server with lower communication and computation cost.

2 Related Work

While more and more data are outsourced to CSP, the data integrity checking has become a necessary technique for secure cloud storage. Over the past decade, two main types of data integrity checking model are proposed. The first is the proof of retrievability (PoR) model [7], which supports the integrity verification for outsourced data and the retrievability of data by error-correcting codes. The second is the PDP model proposed in [8], which supports efficient verification for the data integrity on cloud servers with sampling inspection. Since PDP is considered to be much more flexible and efficient [9], many public PDP schemes have been successively proposed in many literatures to address the data integrity checking problems with various features like data dynamic [10], multi replicas [11-12], privacy preserving [13-14] and so on.

Updating the data in cloud server is a common requirement for data owners especially when data is shared publicly. Therefore, many data integrity checking schemes consider to support the data dynamic feature. For example, Ateniese et al. [15] proposed a limited dynamic scheme based on symmetric cryptography. Erway et al. [16] designed a ranked authenticated skip list by which they gave a fully dynamic data integrity checking scheme. Wang et al. [17] utilized the structure of Merkle Hash Tree to achieve integrity auditing for data dynamic. Tian et al. [18] presented a dynamic scheme relied on a two-dimensional structure called dynamic hash table. Yan et al. [19] employed a hybrid data structure to realize a dynamic PDP scheme for cloud storage. Gudeme et al. [20] based on the certificateless crypto proposed a secure data integrity checking scheme in which an extended double linked list is designed to realize data dynamic operations.

To improve the availability and the security of data in CSP, many clients store the data with multiple replicas, which raise the requirement of the integrity checking for multiple replicas. Curtmola et al. [21] first gave a MR-PDP scheme for integrity verification of multiple replicas in cloud storage. To improve the efficiency, Barsoum et al. [22] presented a multi-copy PDP scheme which supports the public verification and data dynamic. Li et al. [23] proposed an efficient scheme of multi replicas on multiple cloud servers. Peng et al. [24] presented a dynamic data integrity audition scheme for multi-replica which supports batch checking. Yu et al. [25] employed an indexed merkle hash tree to design a dynamic auditing scheme for multi-replica on cloud with geographic location. Zhou et al. [26] constructed a structure named multicopy merkle hash tree, by which they proposed a dynamic multiple replicas data integrity checking scheme.

Data privacy preserving is another security requirement in data integrity checking especially for user's valuable and sensitive data. To protect the data's privacy, Wang et al. [27] designed a public data integrity checking scheme with zero knowledge leakage of the data. To avoid the certificate management, Yu et al. [28] presented an identity-based integrity checking scheme with perfect privacy-preserving of data. Li et al. [29] designed a novel mechanism to hide the data of integrity proof, by which they give a concrete data integrity verification scheme with data privacy-preserving. Tian et al. [30] presented a new scheme with data privacy preserving based on a zero-knowledge proof mechanism. Zhao et al. [31] presented a user stateless data auditing scheme, which protected data's privacy and also realized dynamical operations based on ranked authenticated skip list.

Yu et al. [32] firstly considered key-exposure attack on data integrity checking scheme, they used a binary tree to update client's key with different time periods and offered a practical scheme. However, this scheme is inefficient that the user has to undertake heavy computation cost. Later, Li et al. [33] presented a public integrity auditing scheme with key update and data privacy preserving. Zhang et al. [34] proposed an identity-based public auditing scheme with key-exposure resilient based on lattice assumption, which only achieves the forward security. Xu et al. [35] also presented a key-exposure resilience PDP scheme which realized the backward and forward security of the private key simultaneously. However, this scheme is not a real public scheme and not efficient, because the third party auditor (TPA) should generate private keys to take part in user's key updating, audition authenticator updating and integrity verification. Therefore, it is still an open problem of designing data integrity auditing scheme with the feature of key-exposure resilient.

3 Scheme Definition

3.1 Preliminaries

(1) Bilinear Map: G_1 and G_2 are two multiplicative cyclic groups with order p. g is a generator of G_1. $e: G_1 \times G_1 \to G_2$ is a bilinear pairing if the following three conditions hold:

(a) Bilinearity: $\forall x, y \in G_1$ and $\forall a, b \in Z_p^*$, it has $e(x^a, y^b) = e(x, y)^{ab}$.

(b) Computability: $\forall x, y \in G_1$, the value of $e(x, y)$ can be computed efficiently.

(c) Non-degeneracy: $e(g, g) \neq 1$.

(2) Computational Diffie-Hellman (CDH) Assumption: g is a generator of group G with prime order p, and g^a, g^b are two random elements of G. CDH assumption means that for any adversary A, it is hard to compute g^{ab} within probabilistic polynomial time, in which a, b are unknown. The advantage for solving the CDH problem by the adversary A is negligible, which can be defined as: $Adv_A^{CDH} = Pr[g^{ab} \leftarrow A(g, g^a, g^b)] \leq \varepsilon$.

3.2 System Model

Figure 1 illustrates the system model of the proposed public checking scheme for cloud data with key update, which has the following three entities:

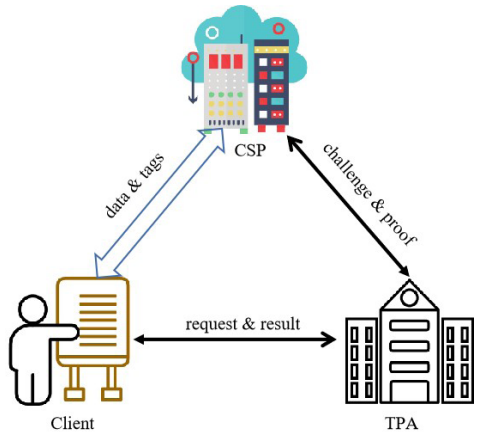

Figure 1. System model

(1) Client, who outsources massive data to cloud server and delegates the integrity checking work to TPA.

(2) CSP, who supplies data storage and management service to clients and responds the integrity checking request from TPA.

(3) TPA, who is trustful and authorized to audit the data integrity on behalf of client, and dedicated to output reliable checking results.

The client prepares all the data which is split to many data blocks. For efficient data checking, client binds an authentication tag for each block. Then client pays for the service of CSP and outsources all the blocks and tags to CSP to decrease the cost of data storage. TPA is a trusted data auditor who is authorized by client to audit the correctness of the data on CSP. CSP is assumed to be semi-honest, that it can provide high-quality services of data storage and maintenance, but may hide data broken incidents to keep his own benefits.

3.3 Outline of the Proposed Scheme

A public cloud data integrity checking scheme with key updating consists of nine algorithms:

(1) *Setup*: before offering the data storage services, CSP executes this algorithm to initialize the global security parameters of the system. The input k is the security value, and *params* is the public parameters.

(2) *KeyGen*: client generates two key pairs by this algorithm, the first one (sk, pk) is used for computing the authentication tags of data blocks. The second one (ssk, spk) is used for generating 'data-tag'.

(3) *TagGen*: client performs this algorithm to generate the authentication tags of blocks. It inputs the data F and client's private keys, outputs a set of block tags and a 'data-tag'.

(4) *KeyUpt*: this algorithm is performed by client to update the private key periodically. It outputs the new private key for client based on the old private key and the current time.

(5) *ReKey*: when client's private key is updated, all the old block tags should be updated too. Client executes this algorithm to generate a re-tag-key which is sent to CSP to update all block tags.

(6) *ReTagGen*: this algorithm is performed by the CSP to update block tags with the tag re-tag-key. By this algorithm all block tags are updated from time period t-1 to time period t.

(7) *Challenge*: TPA generates a verification challenge which is sent to CSP as a data integrity checking demand.

(8) *ProofGen*: after getting the challenge from TPA, CSP outputs a data integrity proof for this challenge by the algorithm.

(9) *Verify*: in this algorithm, the TPA checks the integrity of the cloud data by checking the proof returned from the CSP.

3.4 Security Definition

Completeness is the basic security requirement of the public data integrity checking scheme with key update. Completeness means that when both CSP and TPA performs the scheme honestly, the cloud data should be audited correctly. Specifically, if the CSP stores user's original data well and generates the proof rightly by *ProofGen*, the algorithm *Verify* should output 'true'.

Generally, CSP is thought to be semi-trust, it attempts to conceal the fact of data corruption by launching three attacks: forge attack, replay attack and reply attack. Therefore, the proposed data integrity checking scheme with key update should satisfy the soundness feature which means if the data is broken, CSP cannot generate a forged proof to pass the integrity challenge of TPA. We use a security game to cover the soundness security requirement of the public cloud data integrity checking scheme with key update. The game contains the following phases which involve a challenger C and an adversary A.

(1) Setup: C sets up the system and outputs the public parameters by *Setup*, then generates the key pairs by *KeyGen*. Both the public keys and public parameters are returned to A.

(2) Query: C updates the private key with the period Δt by the *KeyUpt*, and responds the following queries performed by A.

(a) Hash query: A askes for the values of hash functions appeared in the scheme, C returns the values to A.

(b) Secret key query: A adaptively queries the re-tag-key at any time period t, C generates the re-tag-key with the time period t and returns it to A.

(c) Tag query: A randomly selects any blocks in $\{m_1, m_2, m_3, ..., m_n\}$ and sends the blocks to C. C generates the tags for the blocks at time period t and then returns the tags to A.

(3) Challenge: C adaptively chooses a time period t' and the $chal = (c, k_1, k_2)$. C sends all the information to the adversary A and asks A to reply the integrity proof according to the challenge.

(4) Forge: when receiving the challenge, A computes the proof P at time period t' and sends P to C. If P passes the checking by *Verify*, A wins the game.

It is noted that in the game the adversary A should not query the secret keys and the tags of time period t' of the challenged blocks, and the challenged blocks are not necessary to be known by A.

4 Construction of New Scheme

In this section, we give the detailed construction of our new scheme, which consists of the following algorithms.

Setup: CSP sets the security parameter k and chooses G_1 and G_2 of the prime order p. Randomly select a generator g of G_1 and a bilinear map e: $G_1 \times G_1 \rightarrow G_2$. Select three hashes: $h_1\{0,1\}^* \rightarrow G_1$, $h_2\{0,1\}^* \rightarrow G_1$, $h_3\{0,1\}^* \rightarrow Z_p$. Choose a pseudo-random function: ϕ: $Z_p \times Z_p \rightarrow Z_p$ and a pseudo-random permutation φ: $Z_p \times \{1...n\} \rightarrow \{1...n\}$. The public parameters are $params = (p, e, g, G_1, G_2, h_1, h_2, h_3, \phi, \varphi)$.

KeyGen: The client selects a random $a_0 \in Z_p$ as the initial private key $sk_0 = a_0$, and computes the public key $pk = g^{a0}$. Further, client chooses a sign key pair (*ssk*, *spk*).

TagGen: The client splits the data F to n blocks, as $F = (m_1, ..., m_n)$ with each $m_j \in Z_p (j \in [1,n])$. Set the time interval Δt and computes $U = h_2(UID \| \Delta t)$ where UID is the unique identity of the client. Then, randomly selects a value $\lambda \in Z_p$, computes $W = g^{\lambda}$.

For each block m_j, an authentication tag is computed by the following equation (1):

$$\theta_j = (h_1(j \| n) \cdot g^{m_j})^{\lambda} \cdot U^{a_0 t_0}. \quad (1)$$

With the equation (1), the client computes the tag sets $\theta(\theta_1, ..., \theta_n)$. Then client selects a secure signature scheme *SIG* to compute the 'data-tag' with the key *ssk*: $FTag = SIG(FID \| W \| \Delta t \| t_0)$ in which FID is the identity of the data F. Client uploads (F, θ) to CSP, and sends (FID, W, Δt, t_0, $FTag$) to TPA. TPA first checks the validity of $FTag$ by the public key *spk*. If the $FTag$ is valid, TPA stores the (FID, W, Δt, t_0, $FTag$) privately, otherwise TPA asks the client to re-sent these values.

KeyUpt: To resist the key leakage attack, the client updates his private key periodically after generating the initial private key a_0 at time t_0. Assume the time interval of key update is Δt, the current time is t_i which $t_i = t_0 + i*\Delta t$. Then the new private key $sk_i = h_3(t_i)*sk_0 = h_3(t_0 + i*\Delta t)*a_0$.

ReKey: When client's key is updated in each time period, the block tag should be updated too. To decrease the cost of client, this work is outsourced to CSP. Client computes the re-tag-key: $rsk_t = \frac{U^{sk_t \cdot t}}{U^{sk_{t-1} \cdot (t-\Delta t)}}$, by the secret key sk_t, sk_{t-1} and the time t, Δt. Then, client sends the tag-update-key rsk_t to CSP.

ReTagGen: After receiving the rsk_t from the client, CSP generates the new tag for each block by $\theta_{j,t} = \theta_{j,t-1} \cdot rsk_t = (h_1(j \| n) \cdot g^{m_j})^{\lambda} \cdot U^{a_i t}$. Therefore, CSP updates the tags θ_{t-1} to θ_t.

Challenge: At time $t(t > t_0)$, TPA sends an auditing challenge for client's file F to CSP. TPA sets the number c of the challenged blocks and selects random values of $k_1, k_2 \in Z_p$. Then, TPA sends the integrity challenge $chal = (c, k_1, k_2)$ to TPA.

ProofGen: Once receiving the *chal*, TPA obtains the index set of challenged blocks as $V = \{v_i = \varphi(i, k_1) \mid i \in [1,c]\}$, and the random parameters set $S = \{s_i = \phi(i, k_2) \mid i \in [1,c]\}$. TPA computes the proof as: $M = \sum_{i=1}^{c} s_i \cdot m_{v_i}$, $\Gamma = \prod_{i=1}^{c} \theta_{v_i}^{s_i}$. Finally, CSP sends the proof $P = (M, \Gamma)$ to TPA.

Verify: When TPA receives the proof P from CSP, TPA first computes the number of time intervals $l = (t - t_0)/\Delta t, t = l * \Delta t + t_0$ by the start time t_0, time interval Δt and the current time t. Then computes $V = \{v_i = \varphi(i, k_1) \mid i \in [1,c]\}$, and $S = \{s_i = \phi(i, k_2) \mid i \in [1,c]\}$. To verify the data integrity, TPA executes the equation (2).

$$e(\Gamma, g) = e(\prod_{i=1}^{c} h_1(v_i \| n)^{s_i} \cdot g^M, W) \cdot e(U^{\prod_{j=1}^{l} h_3(t_0 + j \cdot \Delta t) \cdot t \cdot \sum_{i=1}^{c} s_i}, pk). \quad (2)$$

If the equation holds, TPA returns "true" to the client. Otherwise, TPA returns "false".

5 Security Analysis

5.1 Completeness Proof

It is easy to see if the Equation (2) is correct, our scheme can verify the cloud data integrity rightly. Based on the properties of bilinear maps, we can prove the rightness of Equation (2) as follows:

$$e(\Gamma, g) = e(\prod_{i=1}^{c} ((h_1(v_i \| n) \cdot g^{m_{v_i}}))^{\lambda} \cdot U^{a_i t})^{s_i}, g)$$

$$= e(\prod_{i=1}^{c} ((h_1(v_i \| n) \cdot g^{m_{v_i}}))^{s_i}, g^{\lambda}) \cdot e(\prod_{i=1}^{c} U^{a_i t s_i}, g)$$

$$= e(\prod_{i=1}^{c} (h_1(v_i \| n)^{s_i}, g^{\lambda}) \cdot e(g^{\sum_{i=1}^{c} m_{v_i} s_i}, g^{\lambda}) \cdot e(U^{a_i t \sum_{i=1}^{c} s_i}, g)$$

$$= e(\prod_{i=1}^{c} (h_1(v_i \| n)^{s_i}, W) \cdot e(g^{\sum_{i=1}^{c} m_{v_i} s_i}, W) \cdot e(U^{\prod_{j=1}^{l} h_3(t_0 + j \cdot \Delta t) a_0 t \sum_{i=1}^{c} s_i}, g)$$

$$= e(\prod_{i=1}^{c} (h_1(v_i \| n)^{s_i} \cdot g^M, W) \cdot e(U^{\prod_{j=1}^{l} h_3(t_0 + j \cdot \Delta t) t \sum_{i=1}^{c} s_i}, pk).$$

5.2 Soundness Proof

Proof: We make use of a series of games to prove our scheme satisfying the soundness property. Let (g, g^a, g^b) be one CDH instance, if the adversary A wins the game, the simulator B can computes g^{ab} with following steps.

Game 0: This game is executed as the definition in section 3.3.

Game 1: Different from game0, in this game, the adversary A stores all the replies return from the challenge B. The detailed processes of this game are as follows.

Setup phase: B chooses all the public parameters, sets g^a to be the public key, and sends all these values to A. Then B selects the start time period t, the time interval Δt, a random value λ and computes the value $W = g^{\lambda}$.

Query phase: A can execute the following three type of queries: hash query, re-key query and tag query.

(1) Hash query: A adaptively sends (j, n) to B for h_1 query. B randomly selects a value $h_1 \in G_1$ and sends h_1 to A. Then B adds an element (j, n, h_1) to the list L_1. Note that if the (j, n) is already in L_1, B directly gets the corresponding h_1 and returns it to A.

A adaptively sends $(Uid, \Delta t)$ to B for h_2 query. B first tosses a coin $\tau \in \{0,1\}$. If $\tau = 0$, B random selects $r \in Z_p$ and returns $h_2 = g^r$ to A, otherwise, B returns $h_2 = (g^b)^r$ to A. Then B adds the element $(Uid, \Delta t, h_2, r, \tau)$ to L_2. Note that if the $(Uid, \Delta t)$ is already in L_2, B directly gets the corresponding h_2 and returns it to A.

A adaptively sends t_i to B for h_3 query. B randomly selects a value $h_3 \in Z_p$ and sends h_3 to A. B adds the element (t_i, h_3) to L_3. Note that if the t_i is already in L_3, B directly gets the corresponding h_3 and returns it to A.

(2) Rekey query: A adaptively sends $(Uid, \Delta t, t)$ to B for querying the re-key for updating the tags. B searches the value (r, r') from L_2 according to $(Uid, \Delta t, t)$. If $\tau = 1$, B aborts and quit the game. Otherwise, B gets the re-tag-key $\frac{g^{ar}t}{g^{ar'(t-\Delta t)}}$.

(3) Tag query: A adaptively sends $(m_j, j, n, Uid, \Delta t, t)$ to B and asks for the tag of m_j at time t under the conditions of Δt and Uid. B finds all the values h_1, h_2, h_3 from the lists of L_1, L_2, L_3. If the corresponding $\tau = 1$, B aborts and quit. Otherwise, B outputs the tag of m_j: $(h_1 \cdot g^{m_j})^{\lambda} \cdot g^{rah_3t} = (h_1 \cdot g^{m_j})^{\lambda} \cdot (g^a)^{rh_3t}$.

Forge phase: A adaptively sends a block $m_{j'}$ and the forged tag $\theta_{j'}$ with the $(j, n, Uid, \Delta t, t)$ to C. If the forged tag can pass the verification, A wins the game. It is noted that A has not queried the tag of $m_{j'}$ under such conditions before, and not queried the tag of the last time period and the re-tag-key of the last time period simultaneously.

Analysis: From the equation (1), it is easy to get that each valid tag can be verified by: $e(\theta_j, g) = e(h_1(j\|n) \cdot g^{m_j}, W) \cdot e(U^{t0}, pk))$.

Therefore, if the $(m_{j'}, \theta_{j'})$ is valid, they should satisfy the above equation too. B searches all the values of (h_1, h_2, h_3) from the lists L_1, L_2, L_3 respectively. If $\tau = 0$, B aborts and return fail. Otherwise, B can get : $e(\theta_{j'}, g) = e(h_1 \cdot g^{m_j}, g^{\lambda}) \cdot e(U^{t0}, g^a) = e((h_1 \cdot g^{m_{j'}})^{\lambda}, g) \cdot e(g^{rbt_0}, g^a) = e((h_1 \cdot g^{m_{j'}})^{\lambda} \cdot g^{brt_0a}, g)$. So, the value of g^{ab} can be computed by: $g^{ab} = \left(\frac{\theta_{j'}}{(h_1 g^{m_{j'}})^{\lambda}}\right)^{1/rt_0}$.

Assume the probability of $\tau = 0$ is γ, so the probability of $\tau = 1$ is $1 - \gamma$. Assume A wins the game with probability δ, after executing ζ_1, ζ_2 times of Re-Key query and Tag query respectively, we can get the probability for B to get the value of g^{ab} is no less than : $\delta \cdot (1-\gamma) \cdot (\gamma)^{\zeta_1 \zeta_2}$. As a result, if the adversary A forges a valid tag with probability of δ, B solves the CDH problem with the probability of $\delta \cdot (1-\gamma) \cdot (\gamma)^{\zeta_1 \zeta_2}$ at least. Therefore, it is computationally impractical for the adversary A to generate a forgery tag in our scheme.

Game 2: Different from Game1, the adversary A submits a complete forged proof $P' = (M', \Gamma')$ of $chal = (c, k_1, k_2)$ to B. We prove that the probability of $P' = (M', \Gamma')$ passing the verification is negligible.

Proof: Assume $P' = (M', \Gamma')$ passes the integrity checking, it easy to get the equation of:

$$e(\Gamma', g) = e(\prod_{i=1}^{c} h_1(v_i \| n)^{s_i} \cdot g^{M'}, W) \cdot e(U^{\prod_{j=1}^{l} h_3(t_0 + j \cdot \Delta t) \cdot t \cdot \sum_{i=1}^{c} s_i}, pk)$$

Let the proof $P = (M, \Gamma)$ is generated by the truthful prover for the challenge $chal = (c, k_1, k_2)$. So the proof P makes the following equation hold:

$$e(\Gamma, g) = e(\prod_{i=1}^{c} h_1(v_i \| n)^{s_i} \cdot g^{M}, W) \cdot e(U^{\prod_{j=1}^{l} h_3(t_0 + j \cdot \Delta t) \cdot t \cdot \sum_{i=1}^{c} s_i}, pk)$$

Observing the two equations, if $\Gamma = \Gamma'$, we can get $g^M = g^{M'}$, i.e. $\sum_{i=1}^{c} s_i m_{v_i} = \sum_{i=1}^{c} s_i m'_{v_i}$, which means each $m_{v_i} = m'_{v_i}$. In this case, the forged proof P' is equal to the truth proof P, which is contrast to the assumption above. If $\Gamma \neq \Gamma'$, there should be $g^M \neq g^{M'}$. In this case, we can forge a tag for $\Delta M = (M - M')$ with $c = 1$, which contrasts to the conclusion of Game1. Therefore, the adversary A forges a valid proof $P' = (M', \Gamma')$ only with negligible probability.

6 Performance Analysis

6.1 Numerical Analysis

First, we show the theoretical analysis of the computation cost of our scheme. Let E_{G_1}, E_{G_2} denote one exponentiation cost on G_1 and G_2 respectively, use P to represent one pairing cost of G_1. The cost of other operations like hash, multiplication and addition is omitted because of their negligible overhead.

To generate a block tag, the client should run the algorithm of *TagGen* which costs $3E_{G_1}$. To resist key-exposure, the client updates his secret key periodically by the *KeyUpt*, which needs one hash and one multiplication cost. After updating the secret key, the client should update all the block tags stored in CSP. The client first generates a re-tag-key by *ReKey* which costs $2E_{G_1}$. With the re-tag-key, CSP updates all the tags by *ReTagGen* which needs only negligible cost. For generating the integrity proof, CSP executes the algorithm *ProofGen*, which costs cE_{G_1}. When TPA receives the proof from CSP, it runs *Verify* to audit the integrity of

data which costs $(c + 2)E_{G_1} + 3P$. Table 1 demonstrates the comparison of our scheme with two recent schemes [33, 35] in terms of computation overhead, where s is the counter of sectors in one block and l is the times of key updating till the challenge performed.

Table 1. Computation cost comparison

Steps	Li's scheme	Xu's scheme	Our scheme
Tag-gen	$(1 + s)\ E_{G_1}$	$3E_{G_1}$	$3E_{G_1}$
Key-update	$2E_{G_1}$	E_{G_1}	-
Tag-update	$2E_{G_1}$	$2E_{G_1}$	$2E_{G_1}$
Proof-gen	$(c + 1)\ E_{G_1} +$ $(s + 1)\ E_{G_1} + 2E_{G_2}$	$cE_{G_1} + E_{G_2}$	cE_{G_1}
Verify	$(c + 3 + s)\ E_{G_1} +$ $(l - 2) \cdot 2P + 5P$	$(c + 4)\ E_{G_1} +$ $3P$	$(c + 2)\ E_{G_1} +$ $3P$

Secondly, we evaluate the communication cost of our scheme where we omit the negligible cost like the size of integer. The re-tag-key for updating tags is sent to CSP by the client, which has the size of $|G_1|$. In the challenge phase, the communication cost for sending the challenge is $2|Z_p|$, and the proof size is $|G_1| + |Z_p|$. Moreover, we show the comparison in terms of communication cost with [28, 30] in the Table 2.

Table 2. Communication cost comparison

Steps	Li's scheme	Xu's scheme	Our scheme
Key-update	——	$\|Z_p\| + \|G_1\|$	——
Re-key	$2\|G_1\|$	$\|G_1\|$	$\|G_1\|$
Challenge	$(c + 2)\ \|Z_p\|$ $+3\ \|G_1\|$	$c\|Z_p\|$	$2\|Z_p\|$
Proof	$(l + s)\ \|Z_p\|$ $+\ l\ \|G_1\|$	$\|Z_p\| + \|G_1\|$ $+\|G_2\|$	$\|G_1\|+ \|Z_p\|$

6.2 Experimental Results

We execute several experiments to test the performance of the proposed scheme. We implement our scheme by C programming language based on PBC library which supplies powerful functions for crypto operations. To make accurately comparisons, the schemes of [33, 35] have been implemented too. All experiments are conducted in ubuntukylin-15.10 operating system with Core i5 CPU and 8G Ram. The elliptic curve of type-A in PBC is used in our experiments with the security parameters of 160 bits. In our experiment, we set n = 100000 data blocks and block size is 2KB. The time interval of key updating is Δt = 10 seconds, the sector number of one block is s = 1 and the times of key updating is l = 10.

Figure 2 shows the cost of the phase of tag generation. In this experiment, we change the counter of blocks from 100 to 1000 so as to observe the cost under each situation. From Figure 2, it is easy to see that all the time cost of tag generation of these three schemes is linearly increasing with the increasing of block number. Xu's scheme has roughly the same cost as ours which needs about 11 seconds to compute 1000 tags. And Li's scheme is more efficient than that of two, which only needs about 7.3 seconds to generate 1000 tags. The experiment results keep consistent with the theoretical analysis above and the truth time for tag generation is very practical. Furthermore, tag generation brings little impact on the performance of data integrity verification because it is done only once.

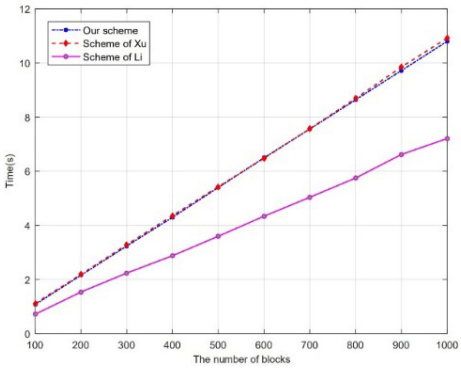

Figure 2. Cost of tag generation

Then we consider the performance of key update which is illuminated in Figure 3. From Figure 3, we can see that the key update cost of all three schemes almost keeps instant no matter how many times the key update. Moreover, our scheme is more efficient than that of other two in this step, that's because in our scheme client's public key is no need to re-generate, there only needs to update the client's secret key which incurs only one hash operation and one multiple operation. Therefore, our scheme consumes only negligible cost.

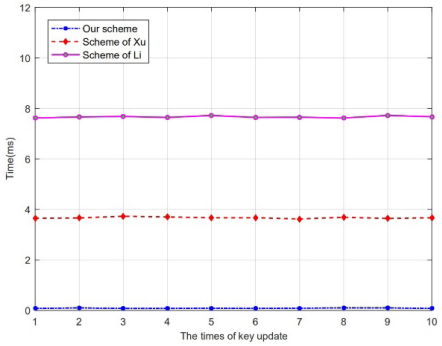

Figure 3. Cost of key update

TPA runs the algorithm of *Verify* to check the integrity of data with the proof returned from CSP. Figure 4 shows the experimental result of proof verification phase of the three schemes. From the Figure 4, we can see that all the three schemes have the linear computation cost with the number of challenged blocks. Li's scheme costs a little more expensive than that of others. Note that here we set the times of key updating to 10, if this value increases, Li's scheme will need more expensive cost because its cost is positive correlation with the times of key updating. However, the verification costs of our scheme and Xu's scheme only depend on the counter of challenged blocks with no relation with the key updating times.

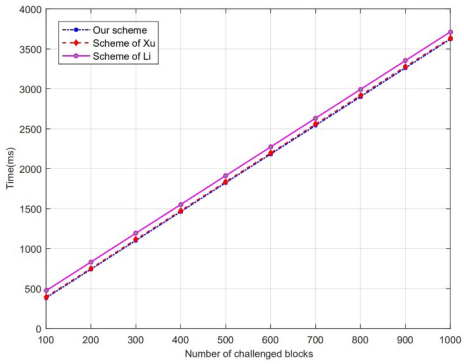

Figure 4. Cost of verification

7 Conclusion

The paper proposes a public data integrity checking scheme for secure cloud storage with efficient key update which achieves the security feature of key exposure resilient. In our scheme, client's key is updated at a fix time period to prevent the key exposure threat, at the same time the block authentication tags are updated too with the client's new key to ensure the data integrity can be verified normally. Moreover, both the key updating and authenticator tag updating of our scheme are very efficient. We give the formal security model of our scheme and prove the security through a serial of security games. Numeric analysis and experimental results demonstrate that our scheme is efficient and practical in comparison with two recent schemes with key update.

Acknowledgments

This work was supported by Program for Scientific Research Foundation for Talented Scholars of Jinling Institute of Technology (JIT-FHXM-202110), the Opening Foundation of Fujian Provincial Key Laboratory of Network Security and Cryptology Research Fund, Fujian Normal University (NSCL-KF2021-02), the National Natural Science Foundation of China (61902163), the Jiangsu Province High Level "333" Program (0401206044).

References

[1] A. A. Abbasi, A. Abbasi, S. Shamshirband, A. T. Chronopoulos, V. Persico, A. Pescapè, Software-defined Cloud Computing: A Systematic Review on Latest Trends and Developments, *IEEE Access*, Vol. 7, pp. 93294-93314, July, 2019.

[2] X. Yan, L. Sun, Z. Sun, J. Zhou, A. Song, Improved Hop-based Localization Algorithm for Irregular Networks, *IET Communications*, Vol. 13, No. 5, pp. 520-527, March, 2019.

[3] X. Yan, J. Cao, L. Sun, J. Zhou, S. Wang, A. Song, Accurate Analytical-Based Multi-Hop Localization With Low Energy Consumption for Irregular Networks, *IEEE Transactions on Vehicular Technology*, Vol. 69, No. 2, pp. 2021-2033, February, 2020.

[4] L. Chen, J. Li, Y. Lu, Y. Zhang, Adaptively Secure Certificate-based Broadcast Encryption and Its Application to Cloud Storage Service, *Information Sciences*, Vol. 538, pp. 273-289, October, 2020.

[5] L. Chen, J. Li, Y. Zhang, Anonymous Certificate-based Broadcast Encryption with Personalized Messages, *IEEE Transactions on Broadcasting*, Vol. 66, No. 4, pp. 867-881, December, 2020.

[6] W. Song, Y. Wu, Y. Cui, Q. Liu, Y. Shen, Z. Qiu, J. Yao, Z. Peng, Public Integrity Verification for Data Sharing in Cloud with Asynchronous Revocation, *Digital Communications and Networks*, Vol. 8, No. 1, pp. 33-43, February, 2022.

[7] A. Juels, B. S. Kaliski Jr., PORs: Proofs of Retrievability for Large Files, *2007, 14th ACM Conference on Computer and Communications Security (CCS)*, Alexandria, Virginia, USA, 2007, pp. 584-597.

[8] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, D. Song, Provable Data Possession at Untrusted Stores, *2007, 14th ACM Conference on Computer and Communications Security (CCS)*, Alexandria, Virginia, USA, 2007, pp. 598-609.

[9] A. Li, Y. Chen, Z. Yan, X. Zhou, S. Shimizu, A Survey on Integrity Auditing for Data Storage in the Cloud: From Single Copy to Multiple Replicas, *IEEE Transactions on Big Data*, Vol. 8, No. 5, pp. 1428-1442, October, 2022.

[10] L. Caruccio, S. Cirillo, V. Deufemia, G. Polese, Efficient Validation of Functional Dependencies during Incremental Discovery, *2021, 29th Italian Symposium on Advanced Database Systems (SEBD)*, Pizzo Calabro, Italy, 2021, pp. 1-12.

[11] L. Li, T. Casalini, P. Arosio, M. Salvalaglio, Modeling the Structure and Interactions of Intrinsically Disordered Peptides with Multiple Replica, Metadynamics-Based Sampling Methods and Force-Field Combinations, *Journal of chemical theory and computation*, Vol. 18, No. 3, pp. 1915-1928, February, 2022.

[12] T. Li, J. Chu, L. Hu, CIA: A Collaborative Integrity Auditing Scheme for Cloud Data with Multi-Replica on Multi-Cloud Storage Providers, *IEEE Transactions on Parallel & Distributed Systems*, Vol. 34, No. 1, pp. 154-162, January, 2023.

[13] F. Cerruto, S. Cirillo, D. Desiato, S. M. Gambardella, G. Polese, Social Network Data Analysis to Highlight Privacy Threats in Sharing Data, *Journal of Big Data*, Vol. 9, No. 1, pp. 1-26, February, 2022.

[14] J. Song, Z. Han, W. Wang, J. Chen, Y. Liu, A New Secure Arrangement for Privacy-preserving Data Collection, *Computer Standards & Interfaces*, Vol. 80, Article No. 103582, March, 2022.

[15] G. Ateniese, R. D. Pietro, L. V. Mancini, G. Tsudik, Scalable and Efficient Provable Data Possession, *2008, 4th International Conference on Security and Privacy in Communication Networks (SecureComm)*, Istanbul, Turkey, 2008, pp. 1-10.

[16] C. Erway, A. Küpçü, C. Papamanthou, R. Tamassia, Dynamic provable data possession, *2009, 16th ACM Conference on Computer and Communications Security (CCS)*, Chicago, Illinois, USA, 2009, pp. 213-222.

[17] Q. Wang, C. Wang, K. Ren, W. Lou, J. Li, Enabling

Public Auditability and Data Dynamics for Storage Security in Cloud Computing, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 22, No. 5, pp. 847-859, May, 2011.
[18] H. Tian, Y. Chen, C. Chang, H. Jiang, Y. Huang, Y. Chen, J. Liu, Dynamic-hash-table Based Public Auditing for Secure Cloud Storage, *IEEE Transactions on Services Computing*, Vol. 10, No. 5, pp. 701-714, September-October, 2017.
[19] H. Yan, J. Li, J. Han, Y. Zhang, A Novel Efficient Remote Data Possession Checking Protocol in Cloud Storage, *IEEE Transactions on Information Forensics and Security*, Vol. 12, No. 1, pp. 78-88, January, 2017.
[20] J. R. Gudeme, S. Pasupuleti, R. Kandukuri, Certificateless Privacy Preserving Public Auditing for Dynamic Shared Data with Group User Revocation in Cloud Storage, *Journal of Parallel and Distributed Computing*, Vol. 156, pp. 163-175, October, 2021.
[21] R. Curtmola, O. Khan, R. Burns, G. Ateniese, MR-PDP: Multiple-replica Provable Data Possession, *2008, 28th International Conference on Distributed Computing Systems (ICDCS)*, Beijing, China, 2008, pp. 411-420.
[22] F. Barsoum, M. A. Hasan, Provable Multicopy Dynamic Data Possession in Cloud Computing Systems, *IEEE Transactions on Information Forensics and Security*, Vol. 10, No. 3, pp. 485-497, March, 2015.
[23] J. Li, H. Yan, Y. Zhang, Efficient Identity-Based Provable Multi-Copy Data Possession in Multi-Cloud Storage, *IEEE Transactions on Cloud Computing*, Vol. 10, No. 1, pp. 356-365, January-March, 2022.
[24] S. Peng, F. Zhou, J. Li, Q. Wang, Z. Xu, Efficient, Dynamic and Identity-based Remote Data Integrity Checking for Multiple Replicas, *Journal of Network and Computer Applications*, Vol. 134, pp. 72-88, May, 2019.
[25] H. Yu, Z. Yang, M. Waqas, S. Tu, Z. Han, Z. Halim, R. O. Sinnott, U. Parampalli, Efficient Dynamic Multi-replica Auditing for the Cloud with Geographic Location, *Future Generation Computer Systems*, Vol. 125, pp. 285-298, December, 2021.
[26] L. Zhou, A. Fu, Y. Mu, H. Wang, S. Yu, Y. Sun, Multicopy Provable Data Possession Scheme Supporting Data Dynamics for Cloud-based Electronic Medical Record System, *Information Sciences*, Vol. 545, pp. 254-276, February, 2021.
[27] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, W. Lou, Privacy-Preserving Public Auditing for Secure Cloud Storage, *IEEE Transactions on Computers*, Vol. 62, No. 2, pp. 362-375, February, 2013.
[28] Y. Yu, M. H. Au, G. Ateniese, X. Huang, W. Susilo, Y. Dai, G. Min, 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, April, 2017.
[29] J. Li, H. Yan, Y. Zhang, Identity-Based Privacy Preserving Remote Data Integrity Checking for Cloud Storage, *IEEE Systems Journal*, Vol. 15, No. 1, pp. 577-585, March, 2021.
[30] H. Tian, F. Nan, C. Chang, Y. Huang, J. Lu, Y. Du, Privacy-preserving Public Auditing for Secure Data Storage in Fog-to-cloud Computing, *Journal of Network and Computer Applications*, Vol. 127, pp. 59-69, February, 2019.
[31] H. Zhao, X. Yao, X. Zheng, T. Qiu, H. Ning, User Stateless Privacy-preserving TPA Auditing Scheme for Cloud Storage, *Journal of Network and Computer Applications*, Vol. 129, pp. 62-70, March, 2019.
[32] J. Yu, K. Ren, C. Wang, V. Varadharajan, Enabling Cloud Storage Auditing with Key-exposure Resistance, *IEEE Transactions on Information Forensics and Security*, Vol. 10, No. 6, pp. 1167-1179, June, 2015.
[33] Y. Li, Y. Yu, B. Yang, G. Min, H. Wu, Privacy Preserving Cloud Data Auditing with Efficient Key Update, *Future Generation Computer Systems*, Vol. 78, pp. 789-798, January, 2018.
[34] X. Zhang, H. Wang, C. Xu, Identity-based Key-exposure Resilient Cloud Storage Public Auditing Scheme from Lattices, *Information Sciences*, Vol. 472, pp. 223-234, January, 2019.
[35] Y. Xu, S. Sun, J. Cui, H. Zhong, Intrusion-resilient Public Cloud Auditing Scheme with Authenticator Update, *Information Sciences*, Vol. 512, pp. 616-628, February, 2020.

Biographies

Hao Yan, Ph.D., he is currently an assistant professor with the School of Network Security, Jinling Institute of Technology, Nanjing, China. His research interests include cryptography and information security, cloud computing, network security and trusted computing etc.

Yanan Liu, Ph.D., she is currently an assistant professor with the School of Network Security, Jinling Institute of Technology, Nanjing, China. Her research interests include Information security and Applied Cryptography, especially includes Key Management in WSNs, Secure Communication in IoT, PUF-based authentication, etc. authentication, etc.

Dandan Huang is an associate professor with the School of Network Security at Jinling Institute of Technology. She received her Ph.D. degree in Applied Mathematics from University of Chinese Academy of Sciences. Her research interests include Information Security and Cryptography etc.

Shuo Qiu is a lecturer in the School of Network Security at Jinling Institute of Technology. She received her Ph.D. degree in the School of Computer and Information Technology at Beijing Jiaotong University in 2017. Her research interests are in cryptographic protocols, big data security and privacy preserving.

Zheng Zhang received a B.S. degree in applied mathematics from the Shanghai Jiao Tong University. He is currently a Research Fellow with the School of Network Security, Jinling Institute of Technology, Nanjing, China, and a Ph.D. student of Southeast University. His research interests include cryptography, secure protocol, network security, etc.