Enabling Cloud Storage Auditing With Verifiable Outsourcing of Key Updates

J. Sridhar*1 M. Swapna*2
*1Associate Professor Department of CSE, *2M.Tech
*Nalla Narasimha Reddy Educational Society’s Group of Institution

Abstract—Key-exposure resistance has always been an important issue for in-depth cyber defence in many security applications. Recently, how to deal with the key exposure problem in the settings of cloud storage auditing has been proposed and studied. To address the challenge, existing solutions all require the client to update his secret keys in every time period, which may inevitably bring in new local burdens to the client, especially those with limited computation resources, such as mobile phones. In this paper, we focus on how to make the key updates as transparent as possible for the client and propose a new paradigm called cloud storage auditing with verifiable outsourcing of key updates. In this paradigm, key updates can be safely outsourced to some authorized party, and thus the key-update burden on the client will be kept minimal. In particular, we leverage the third party auditor (TPA) in many existing public auditing designs, let it play the role of authorized party in our case, and make it in charge of both the storage auditing and the secure key updates for key-exposure resistance. In our design, TPA only needs to hold an encrypted version of the client’s secret key while doing all these burdensome tasks on behalf of the client. The client only needs to download the encrypted secret key from the TPA when uploading new files to cloud. Besides, our design also equips the client with capability to further verify the validity of the encrypted secret keys provided by the TPA. All these salient features are carefully designed to make the whole auditing procedure with key exposure resistance as transparent as possible for the client. We formalize the definition and the security model of this paradigm. The security proof and the performance simulation show that our detailed design instantiations are secure and efficient.

Keywords—Cloud storage, outsourcing computing, cloud storage auditing, key update, verifiability.

I. INTRODUCTION

CLOUD computing, as a new technology paradigm with promising further, is becoming more and more popular nowadays. It can provide users with seemingly unlimited computing resource. Enterprises and people can outsource time-consuming computation workloads to cloud without spending the extra capital on deploying and maintaining hardware and software. In recent years, outsourcing computation has attracted much attention and been researched widely. It has been considered in many applications including scientific computations [1], linear algebraic computations [2], linear programming computations [3] and modular exponentiation computations [4], etc. Besides, cloud computing can also provide users with seemingly unlimited storage resource. Cloud storage is universally viewed as one of the most important services of cloud computing. Although cloud stor-age provides great benefit to users, it brings new security challenging problems. One important security problem is how to efficiently check the integrity of the data stored in cloud. In recent years, many auditing protocols for cloud storage have been proposed to deal with this problem. These pro-tocols focus on different aspects of cloud storage auditing such as the high efficiency [5]–[17], the privacy protection of data [18], the privacy protection of identities [19], dynamic data operations [13], [15], [16], [20], the data sharing [21], [22], etc. The key exposure problem, as another important problem in cloud storage auditing, has been considered [23] recently. The problem itself is non-trivial by nature. Once the client’s secret key for storage auditing is exposed to cloud, the cloud is able to easily hide the data loss incidents for maintaining its reputation, even discard the client’s data rarely accessed for saving the storage space. Yu et al. [23] constructed a cloud storage auditing protocol with key-exposure resilience by updating the user’s secret keys periodically. In this way, the damage of key exposure in cloud storage auditing can be reduced. But it also brings in new local burdens for the client because the client has to execute the key update algorithm in each time period to make his secret key move forward. For some clients with limited computation resources, they might not like doing such extra computations by themselves in each time period. It would be obviously more attractive to make key updates as transparent as possible for the client, especially in frequent key update scenarios. In this
paper, we consider achieving this goal by outsourcing key updates. However, it needs to satisfy several new requirements to achieve this goal. Firstly, the real client’s secret keys for cloud storage auditing should not be known by the authorized party who performs outsourcing computation for key updates. Otherwise, it will bring the new security threat. So the authorized party should only hold an encrypted version of the user’s secret key for cloud storage auditing. Secondly, because the authorized party performing outsourcing computation only knows the encrypted secret keys, key updates should be completed under the encrypted state. In other words, this authorized party should be able to update secret keys for cloud storage auditing from the encrypted version he holds. Thirdly, it should be very efficient for the client to recover the real secret key from the encrypted version that is retrieved from the authorized party. Lastly, the client should be able to verify the validity of the encrypted secret key after the client retrieves it from the authorized party. The goal of this paper is to design a cloud storage auditing protocol that can satisfy above requirements to achieve the outsourcing of key updates. The main contributions are as follows: (1) We propose a new paradigm called cloud storage auditing with verifiable outsourcing of key updates. In this new paradigm, key-update operations are not performed by the client, but by an authorized party. The authorized party holds an encrypted secret key of the client for cloud storage auditing and updates it under the encrypted state in each time period. The client downloads the encrypted secret key from the authorized party and decrypts it only when he would like to upload new files to cloud. In addition, the client can verify the validity of the encrypted secret key. 

II. RELATED WORK

Outsourcing Computation: How to effectively outsource time-consuming computations has become a hot topic in the research of the theoretical computer science in the recent two decades. Outsourcing computation has been considered in many application domains. Chaum and Pedersen [24] firstly proposed the notion of wallet databases with observers, in which a hardware was used to help the client perform some expensive computations. The method for secure outsourcing of some scientific computations was proposed by Atallah et al. [1]. Chevallier-Mames et al. [25] designed the first effective algorithm for secure delegation of elliptic-curve pairings based on an untrusted server. The first out-sourcing algorithm for modular exponentiations was proposed by Hohenberger and Lysyanskaya [26], which was based on the methods of precomputation and server-aided compu-tation. Atallah and Li [27] proposed a secure outsourcing algorithm to complete sequence comparisons. Chen et al. [29] proposed new algorithms for secure outsourcing of modular exponentiations. Benjamin and Atallah [2] researched on how to securely outsource the computation for linear algebra. Atallah and Frikken [28] gave further improvement based on the weak secret hiding assumption. Wang et al. [3] presented an efficient method for secure outsourcing of linear programming computation. Chen et al. [29] proposed an outsourcing algorithm for attribute-based signatures computations. Zhang et al. [30] proposed an efficient method for outsourcing a class of homomorphic functions.

Cloud Storage Auditing: How to check the integrity of the data stored in cloud is a hot topic in cloud security. The notion of “provable data possession” (PDP) was firstly proposed by Ateniese et al. [5] to ensure data possession at untrusted servers. The notion of “proof of retrievability” (PoR) was proposed by Juels et al. [6] to ensure both possession and retriev-ability of data at untrusted servers. Wang et al. [18] proposed a public privacy-preserving auditing protocol. They used the random masking technique to make the protocol achieve privacy-preserving property. Proxy provable data possession protocol was proposed in [17]. The auditing protocols supporting dynamic data operations were also proposed in [13] and [20]. Yang and Jia [16] proposed an auditing protocol supporting both the dynamic property and the privacy preserving property. The privacy preserving of the user’s identity for shared data auditing was considered in [19]. The problem of user revocation in shared data auditing was considered in [21]. Yuan and Yu [22] proposed a public auditing protocol for data sharing with multiuser modification. Sookhak et al. [31] proposed a public cloud auditing protocol for securing big data storage based on algebraic signature. Guan et al. [32] proposed the first cloud storage auditing protocol based on indistinguishability obfuscation, which is especially useful for low-power cloud users. Yang et al. [33] proposed a public auditing protocol for shared cloud data supporting both iden-tity privacy and identity traceability. All above auditing protocols are all built on the assumption that the secret key of the client is absolutely secure and would not be exposed. In [23], the authors firstly considered the key exposure problem in cloud storage auditing and proposed a cloud storage auditing protocol with key-exposure resilience. In that protocol, the secret keys for cloud storage auditing are updated periodically. As a result,
any dishonest behaviors, such as deleting or modifying the client’s data previously stored in cloud, can all be detected, even if the cloud gets the client’s current secret key for cloud storage auditing. However, the client needs to update his secret key in each time period. It will add obvious computation burden to the client, especially when key updates are very frequent.

B. Organization
The rest paper is organized as follows: In Section 2, we introduce the system model, definitions, and preliminaries of our work. Then, we give a concrete description of our protocol in Section 3. Security and performance are analyzed in Section 4. We conclude the paper in Section 5. The detailed security proof is shown in the appendix.

III. MODEL, DEFINITIONS AND PRELIMINARIES

A. Model
We show the system model for cloud storage auditing with verifiable outsourcing of key updates in Fig. 1. There are three parties in the model: the client, the cloud and the third-party auditor (TPA). The client is the owner of the files that are uploaded to cloud. The total size of these files is not fixed, that is, the client can upload the growing files to cloud in different time points. The cloud stores the client’s files and provides download service for the client. The TPA plays two important roles: the first is to audit the data files stored in cloud for the client; the second is to update the encrypted secret keys of the client in each time period. The TPA can be considered as a party with powerful computational capability or a service in another independent cloud. Similar to [23], the whole lifetime of the files stored in cloud is divided into \( T + 1 \) time periods (from 0-th to \( T \)-th time periods). Each file is assumed to be divided into multiple blocks. In order to simplify the description, we do not furthermore divide each block into multiple sectors [7] in the description of our protocol. In the end of each time period, the TPA updates the encrypted client’s secret key for cloud storage auditing according to the next time period. But the public key keeps unchanged in the whole time periods. The client sends the key requirement to the TPA only when he wants to upload new files to cloud. And then the TPA sends the encrypted secret key to the client. After that, the client decrypts it to get his real secret key, generates authenticators for files, and uploads these files along with authenticators to cloud. In addition, the TPA will audit whether the files in cloud are stored correctly by a challenge-response protocol between it and the cloud at regular time.

B. Definitions

(1) The definition of cloud storage auditing protocol with verifiable outsourcing of key updates.

Definition 1: A cloud storage auditing protocol with secure outsourcing of key updates is composed by seven algorithms (SysSetup, EkeyUpdate, VerESK, DecESK, AuthGen, Proof-Gen, ProofVerify), shown below:

1) SysSetup: the system setup algorithm is run by the client. It takes as input a security parameter \( k \) and the total number of time periods \( T \), and generates an encrypted initial client’s secret key \( ESK_0 \), a decryption key \( DK \) and a public key \( PK \). Finally, the client holds \( DK \), and sends \( ESK_0 \) to the TPA.

2) EkeyUpdate: the encrypted key update algorithm is run by the TPA. It takes as input an encrypted client’s secret key \( ESK_j \), the current period \( j \) and the public key \( PK \), and generates a new encrypted secret key \( ESK_{j+1} \) for period \( j + 1 \).

3) VerESK: the encrypted key verifying algorithm is run by the client. It takes as input an encrypted client’s secret key \( ESK_j \), the current period \( j \) and the public key \( PK \), and generates a new encrypted secret key \( ESK_{j+1} \) for period \( j + 1 \).

4) DecESK: the secret key decryption algorithm is run by the client. It takes as input an encrypted client’s secret key \( ESK_j \), the current period \( j \) and the public key \( PK \), and generates a new decrypted secret key \( SK_j \) in this time period.

5) AuthGen: the authenticator generation algorithm is run by the client. It takes as input a file \( F \), a client’s secret key \( SK_j \), the current period \( j \) and the public key \( PK \), and generates a new encrypted secret key \( ESK_j \).
key $P_K$, and generates the set of authenticators for $F$ in time period $j$.

6) **Proof Gen:** the proof generation algorithm is run by the cloud. It takes as input a file $F$, a set of authenticators, a challenge $Chal$, a time period $j$ and the public key $P_K$, and generates a proof $P$ which proves the cloud stores $F$ correctly.

7) **Proof Verify:** the proof verifying algorithm is run by the TPA. It takes as input a proof $P$, a challenge $Chal$, a time period $j$, and the public key $P_K$, and returns “True” if $P$ is valid; or “False”, otherwise.

(2) **Definition of Security**

As same as other cloud storage auditing protocols [5]–[7], [9]–[13], [15]–[18], [20], the malicious cloud is viewed as the adversary in our security model. We use three games (Game 1, Game 2 and Game 3) to describe the adversaries with different compromising abilities who are against the security of the proposed protocol. Specifically, Game 1 describes an adversary, who fully compromises the TPA to get all encrypted secret keys $E_SK_j$ (periods $j = 0, \ldots, T$), tries to forge a valid authenticator in any time period. This game, in fact, shows the security should satisfy that the TPA cannot help the cloud to forge any authenticator in any time period even if it knows the encrypted secret keys. Game 2 describes an adversary, who compromises the cloud to get $D_K$, tries to forge a valid authenticator in any time period. This game, in fact, shows the security should satisfy that an adversary cannot forge any authenticator in any time period even if it gets the decryption secret key $D_K$ by attacking the client. Game 3 provides the adversary more abilities, which describes an adversary, who compromises the client and the TPA to get both $E_SK_j$ and $D_K$ at one time period, tries to forge a valid authenticator in any time period.

Game 1 is composed of four phases.

1) **Setup phase.** The challenger runs the system setup algorithm to generate the initial encrypted client’s secret key $E_SK0$, the decryption key $D_K$ and the public key $P_K$. The challenger updates the encrypted keys to get $E.SK1, \ldots, E.SKT$ by running $E.keyUpdate$ algorithm. The challenger sends $E_SK0, \ldots, E.SKT$ and $P_K$ to an adversary $A$, and holds $D_K$ himself. Set time period $j = 0$.

2) **Query phase.** A running in his phase can adaptively query the authenticators of the blocks specified by it in time period $j$ as follows.

   It adaptively selects and sends a series of blocks $m_1, \ldots, m_n$ to the challenger. The challenger runs $DecE$ to decrypt $E_SK_j$, computes the authenticators for $mi (i = 1, \ldots, n)$ in time period $j$, and sends these authenticators to $A$. $A$ stores these blocks and authenticators. Set time period $j = j + 1$.

At the end of each time period, $A$ can select to stay in this phase or go to the next phase.

3) **Challenge phase.** The challenger sends the adversary $A$ a challenge $Chal$ and a time period $j^*$. The adversary is required to produce a proof of possession for the blocks $m_{s_1}, \ldots, m_{s_c}$ of file $F = (m_1, \ldots, m_n)$ under $Chal$ in time period $j^*$, where $1 \leq s_i \leq n$, $1 \leq l \leq c$, and $1 \leq c \leq n$.

4) **Forgery phase.** $A$ outputs a proof of possession $P$ for the blocks indicated by $Chal$ in time period $j^*$. If $ProofVerify(P_K, j^*, Chal, P) = \text{"True"}$, then $A$ wins in above game.

Game 2 is similar to the Game 1 except that the Setup phase is a little different. In the Setup phase of Game 2, the challenger needs to provide $A$ with $D_K$ and $P_K$ instead of $E_SK0, \ldots, E.SKT$ and $P_K$ in Game 1. Specifically, Game 2 is also composed of four phases.

1) **Setup phase.** The challenger runs the system setup algorithm to generate the initial encrypted client’s secret key $E_SK0$, the decryption key $D_K$ and the public key $P_K$. The challenger sends $D_K$ and $P_K$ to an adversary $A$, and holds $E_SK0$ himself. Set time period $j = 0$.

   The phases 2)-4) are the same as those in Game 1. Game 3 is different from Game 1 and Game 2, in which the adversary is allowed to get encrypted secret key and decryption key simultaneously at a certain time period. Specifically, Game 3 is composed of five phases.

   1) **Setup phase.** The challenger runs the system setup algorithm to generate the initial encrypted client’s
secret key E S K0, the decryption key D K and the public key P K. The challenger sends P K to an adversary A, and holds D K and E S K0 himself. Set time period j = 0.

2) Query phase. As same as Game 1.

3) Break-in phase. Set the break-in time period b = j. The challenger sends E S Kb and D K to the adversary. It means A can execute Dec E S K algorithm to recover the secret key S Kb.

4) Challenge phase. The challenger sends the adversary A a challenge Chal and a time period j ∗ (j ∗ < b). The adversary is required to produce a proof of possession for the blocks ms1 , · · · , msc of file F = (m1 , · · · , mn ) under Chal in time period j ∗, where 1 ≤ sl ≤ n, 1 ≤ l ≤ c, and 1 ≤ c ≤ n.

5) Forgery phase. A outputs a proof of possession P for the blocks indicated by Chal in time period j ∗. If ProofVerify(P K , j ∗ , Chal, P) = “True”, then A wins in above game.

The above security model formalizes the adversaries with different reasonable abilities who try to cheat the challenger that he owns one file he in fact does not entirely know. We give Definition 2 to show that knowledge extractors can extract the challenged file blocks when adversaries output valid proofs in above games. Definition 3 describes the detectability for cloud storage auditing protocol, which shows that the cloud actually keeps the blocks that are not challenged with high probability. Definition 4 describes the verifiability for cloud storage auditing protocol, which shows that the client can verify whether the TPA provides the well-formed encrypted secret key.

Definition 2: We say a cloud storage auditing protocol with verifiable outsourcing of key updates is secure if the adversary A in above games that can cause the challenger to accept its proof with non-negligible probability, there exists efficient knowledge extractors that can extract the challenged file blocks except possibly with negligible probability.

Definition 3: A cloud storage auditing protocol with verifiable outsourcing of key updates is called a (ρ , δ) detectable protocol (0 < ρ , δ < 1) if, given a fraction ρ of corrupted blocks, the probability that the corrupted blocks are found is at least δ.

Definition 4: We say the cloud storage auditing protocol with outsourcing of key updates is verifiable if the valid encrypted secret keys provided by the TPA will always pass the client’s verification, while the invalid ones will always be detected.

IV. CONCLUSION

In this paper, we study on how to outsource key updates for cloud storage auditing with key-exposure resilience. We propose the first cloud storage auditing protocol with verifiable outsourcing of key updates. In this protocol, key updates are outsourced to the TPA and are transparent for the client. In addition, the TPA only sees the encrypted version of the client’s secret key, while the client can further verify the validity of the encrypted secret keys when downloading them from the TPA. We give the formal security proof and the performance simulation of the proposed scheme.

REFERENCES


