IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 21, 2018, accepted April 30, 2018, date of publication May 15, 2018, date of current version June 5, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2836328

Efficient Client-Side Deduplication of Encrypted Data With Public Auditing in Cloud Storage

TAEK-YOUNG YOUN[1], KU-YOUNG CHANG[1], KYUNG-HYUNE RHEE[2], AND SANG UK SHIN[2], (Member, IEEE)

[1]Electronics and Telecommunications Research Institute, Daejeon 34129, South Korea
[2]Department of IT Convergence and Application Engineering, Pukyong National University, Busan 48513, South Korea

Corresponding author: Sang Uk Shin (shinsu@pknu.ac.kr)

This work was supported in part by the Electronics and Telecommunications Research Institute Grant through the Korean Government (Core Technology Research on Trust Data Connectome) under Grant 18ZH1200, in part by the Institute for Information and Communications Technology Promotion (IITP) Grant through the Korean Government (MSIT) (Development of Cyber Self Mutation Technologies for Proactive Cyber Defence) under Grant 2017-0-00213, and in part by the IITP Grant through the Korean Government (MSIT) (the Development of a Secure Framework and Evaluation Method for Blockchain) under Grant 2017-0-00156.

ABSTRACT At present, there is a considerable increase in the amount of data stored in storage services, along with dramatic evolution of networking techniques. In storage services with huge data, the storage servers may want to reduce the volume of stored data, and the clients may want to monitor the integrity of their data with a low cost, since the cost of the functions related to data storage increase in proportion to the size of the data. To achieve these goals, secure deduplication and integrity auditing delegation techniques have been studied, which can reduce the volume of data stored in storage by eliminating duplicated copies and permit clients to efficiently verify the integrity of stored files by delegating costly operations to a trusted party, respectively. So far many studies have been conducted on each topic, separately, whereas relatively few combined schemes, which support the two functions simultaneously, have been researched. In this paper, we design a combined technique, which performs both secure deduplication of encrypted data and public integrity auditing of data. To support the two functions, the proposed scheme performs challenge-response protocols using the BLS signature-based homomorphic linear authenticator. We utilize a third party auditor for performing public audit, in order to help low-powered clients. The proposed scheme satisfies all the fundamental security requirements. We also propose two variances that provide higher security and better performance.

INDEX TERMS Cloud storage, cryptography, data security, information security, public audit, secure deduplication.

I. INTRODUCTION

In cloud storage services, clients outsource data to a remote storage and access the data whenever they need the data. Recently, owing to its convenience, cloud storage services have become widespread, and there is an increase in the use of cloud storage services. Well-known cloud services such as Dropbox and iCloud are used by individuals and businesses for various applications. A notable change in information-based services that has happened recently is the volume of data used in such services due to the dramatic evolution of network techniques. For example, in 5G networks, gigabits of data can be transmitted per second, which means that the size of data that is dealt by cloud storage services will increase due to the performance of the new networking technique. In this viewpoint, we can characterize the volume of data as a main feature of cloud storage services. Many service providers have already prepared high resolution contents for their service to utilize faster networks. For secure cloud services in the new era, it is important to prepare suitable security tools to support this change.

Larger volumes of data require higher cost for managing the various aspects of data, since the size of data influences the cost for cloud storage services. The scale of storage should be increased according to the quantity of data to be stored. In this viewpoint, it is desirable for storage servers to reduce the volume of data, since they can increase their profit by reducing the cost for maintaining storage. On the other hand, clients are mainly interested in the integrity of their

data stored in the storage maintained by service providers. To verify the integrity of stored files, clients need to perform costly operations, whose complexity increases in proportion to the size of data. In this viewpoint, clients may want to verify the integrity with a low cost regardless of the size of data. Owing to the demands of storage servers and clients, many researches on this topic are available in the literature.

To reduce the volume of data, deduplication has to be performed in servers so that the storage space efficiency can be improved by removing duplicated copies. According to the research report of EMC, about 75% of the data are duplicated [7]. This fact raises the need for design of deduplication technology. In the literature, there are studies on two types of deduplication techniques. Among them, client-side deduplication has attracted the interest of researchers more than server-side deduplication due to its efficiency in computation and communication. Unfortunately, client-side deduplication has a number of problems. In [9], Harnik *et al.* discovered some attack scenarios related to client-side deduplication, which can lead to data leakage in the worst case. So far, many schemes and techniques have been introduced to support secure deduplication. In [8], Halevi *et al.* introduced the concept of proofs of ownership (PoW). Keelveedhi *et al.* [11] formalized a class of message-locked encryptions including an existing convergent encryption (CE), and presented a new deduplication technique called DupLESS which is the first deduplication mechanism that can ensure semantic security.

When clients use cloud storage services, the integrity of stored data is the most important requirement. In other words, clients want to be guaranteed about the integrity of their data in the cloud. In cloud storage services, we cannot exclude the possibility of weak cloud servers, which are vulnerable to internal and external security threats. In the case of data loss due to some incident, weak servers may try to hide the fact that they lost some data, which were entrusted by their clients. More seriously, servers delete rarely accessed users' data in order to increase the profit. Therefore, it is a natural requirement of clients to periodically check the current state of their data. To do this in practice, we need a way to efficiently check the integrity of data in remote storage. So far, various schemes have been proposed including proof of retrievability (POR) schemes [4], [10], [14], [17] and provable data possession (PDP) schemes [1], [2], [6], [15].

Secure deduplication and integrity auditing are fundamental functions required in cloud storage services. Hence, individual researches have been actively conducted on these two topics. However, relatively few studies have been conducted for designing a combined scheme that can support these two functions at the same time. The fundamental goal of the design of a combined model is to guarantee less overhead than a trivial combination of existing schemes. In particular, the goal of this paper is to improve the cost of both computation and communication.

In this paper, we design a new scheme for secure and efficient cloud storage service. The scheme supports both secure deduplication and integrity auditing in a cloud environment. In particular, the proposed scheme provides secure deduplication of encrypted data. Our scheme performs PoW for secure deduplication and integrity auditing based on the homomorphic linear authenticator (HLA), which is designed using BLS signature. The proposed scheme also supports public auditing using a TPA (Third Party Auditor) to help low-powered clients. The proposed scheme satisfies all fundamental security requirements, and is more efficient than the existing schemes that are designed to support deduplication and public auditing at the same time. Note that the preliminary version of this paper appeared in MobiSec2017 [16]. The main improvement in this paper is that we propose two variations to provide higher security and better performance. In the first variance, which is designed for stronger security, we assume a stronger adversary and provide a counter measure against the adversary. In the second variance, we design a technique that supports a very low-powered client and entrusts more computation to the cloud storage server in the upload procedure.

This paper is organized as follows. Section II describes related works. In Section III, we propose a secure deduplication technique, which supports integrity auditing based on BLS signature, and analyze it in Section IV. In addition, we suggest two improved protocols in Section V. Finally, Section VI provides the conclusion.

II. RELATED WORKS

Secure deduplication is interesting for both industrial and research communities; therefore, several secure deduplication schemes have been proposed. Harnik *et al.* [9] showed some attacks in the case of client-side deduplication, which causes data leakage. To counter the attacks, the concept of PoW was introduced in [8]. Later, in [11], the convergent encryption, which is defined as message-locked encryption, was formalized and then, Bellare *et al.* presented another scheme called DupLESS for semantic security.

To support data integrity, two concepts, PDP and POR, have been introduced. Ateniese *et al.* [1] introduced PDP for ensuring that the cloud storage providers actually possess the files without retrieving or downloading the entire data. It is basically a challenge-response protocol between the verifier (a client or TPA) and the prover (a cloud). Compared to PDP, POR not only ensures that the cloud servers possess the target files, but also guarantees their full recovery [10]. Since then, a number of POR schemes [4], [14], [17] and PDP schemes [2], [6], [15] have been proposed.

A simple combination of two independent techniques designed for the two above mentioned issues does not efficiently deal with the issues at once, because achieving storage efficiency contradicts with the deduplication of authentication tags. In [18], public auditing with a deduplication scheme based on homomorphic linear authentication tags was proposed. Each user has to generate the integrity tags, even for the file in the cloud. Moreover, the file is available in its plain form on the cloud side. Li *et al.* [12] proposed an integrity auditing scheme for encrypted deduplication storage. This

scheme is based on homomorphic verifiable tags and Merkle hash tree. A user encrypts his file by using a convergent encryption technique and uploads the file to a fully trusted TPA.

III. THE PROPOSED SCHEME

Here, we describe the system model of our scheme. We also give the corresponding security model. After that, we will give a detailed description of our scheme according to the models.

A. SYSTEM AND SECURITY MODEL

Our scheme utilizes the BLS signature-based Homomorphic Linear Authenticator (HLA), which was proposed in [14], for integrity auditing and secure deduplication. We also introduce TPA to support public integrity auditing. The proposed scheme consists of the following entities.

- Client (or user).
  Outsources data to a cloud storage. CE-encrypted data is first generated, and then uploaded to the cloud storage to protect confidentiality. The client also needs to verify the integrity of the outsourced data. To do this, the client delegates integrity auditing to the TPA.
- Cloud Storage Server (CSS).
  Provides data storage services to users. Deduplication technology is applied to save storage space and cost. We consider that the CSS may act maliciously due to insider/outsider attacks, software/hardware malfunctions, intentional saving of computational resources, etc [13]. During the deduplication process, the CSS carries out the PoW protocol to verify that the client owns the file. Moreover, in the integrity audit process, it is necessary to generate and respond to a proof corresponding to the request of the TPA.
- TPA (Third Party Auditor).
  Performs integrity auditing on behalf of the client to reduce the client's processing cost. Instead of the client, the auditor sends a challenge to the storage server to periodically perform an integrity audit protocol. TPA is assumed to be a semi-trust model, that is, an honest but curious model. Under the assumption, it is assumed that the TPA does not collude with other entities.

The relation between entities can be seen in Fig. 1. A client and a CSS perform PoW for secure deduplication, and a TPA is placed between the client and the CSS to execute integrity auditing instead of the client.

Here, we consider the following types of adversary models: outside adversary, insider adversary CSS, and semi-honest adversary TPA.

- Outside adversary: Assuming that the communication channel is not secure, an outside attacker can easily intercept the transmitted data. An outside attacker attempts to pass the PoW process as if it were the proper owner of the data.
- Insider adversary CSS: The CSS assumes that it can act maliciously. It attempts to get information out of the user's encrypted data, and modify or delete the user's data.
- Semi-honest adversary TPA: The TPA is assumed to perform the protocol correctly; however, in the process it tries to obtain information about the user's data.

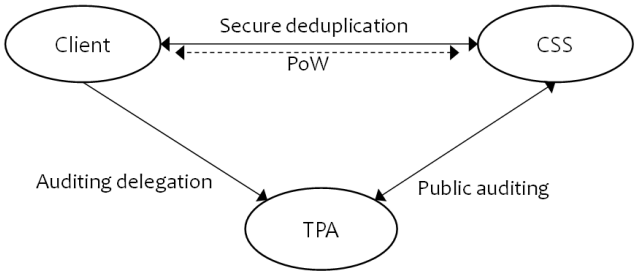


FIGURE 1. System model.

In addition, the proposed scheme should satisfy the following security objectives.

- Privacy: Except for the information about duplication, no information about the outsourced data is revealed to an adversarial party.
- Secure deduplication: Secure deduplication is supported without revealing any information except for the information about duplication.
- Public verifiability: The TPA is able to examine the accuracy and availability of the outsourced data without querying the entire data and without intervention by the data owner.
- Storage correctness: If the CSS is keeping the user's data intact, it can pass the TPA's verification.

B. DETAILED OPERATION OF THE PROPOSED METHOD

The proposed scheme does not compute authentication values separately for a proof of the PoW process and for a proof of the integrity auditing; instead, it computes only one authentication value depending on the duplication. The proposed scheme uses the BLS signature based homomorphic authenticator [14] to generate the authentication value to provide secure deduplication and public integrity auditing. Let $e : G \times G \rightarrow G_T$ be a computable bilinear map with group G's support being $\mathbb{Z}_p$ for some large prime p. g is a generator of G, and $BLSHash : \{0,1\}^* \rightarrow G$ is the BLS hash [3]. A user chooses a random $\alpha \xleftarrow{R} \mathbb{Z}_p$, and computes $v = g^{\alpha} (\in G)$. The user's private key is $sk = \alpha$, and the public key is $pk = v$. The user also generates (spk, ssk) to digitally sign a file using a cryptographically secure signature scheme such as RSA PSS, or DSA. It is assumed that the public key is distributed securely to the entities.

The proposed method includes the following three procedures.

- First upload procedure: In this case, a user first uploads a file that is not stored in the CSS. First, a file ID/Tag and a convergent encryption key K are generated, and

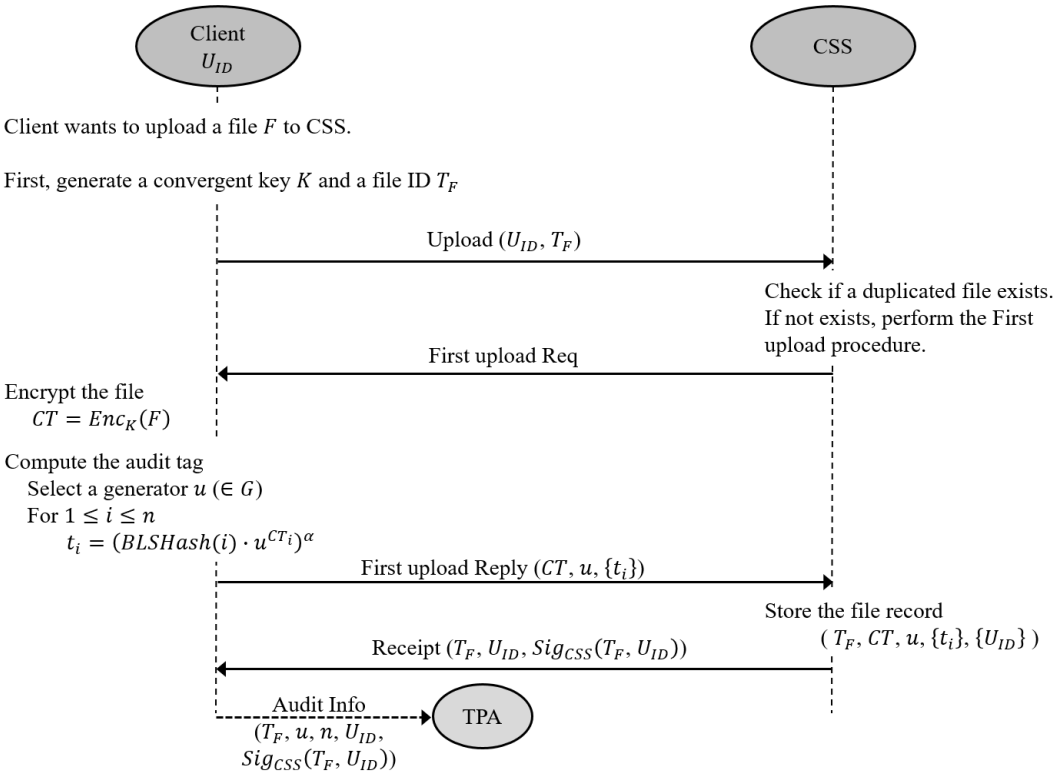


FIGURE 2. First upload procedure.

the file is encrypted using CE with K and then uploaded to the CSS. The CSS maintains the list of owner, tag, and ciphertext. The user computes an authentication tag for the integrity auditing and sends it to the TPA.

- Subsequent upload procedure: This procedure is performed when a duplicate file is uploaded. The CSS checks for the duplication using the file tag, and in the event of duplication, it proceeds with the PoW protocol to examine the ownership of the user. If a user passes this process, the CSS adds the file ownership of the user to the stored file.
- Integrity auditing procedure: Periodic auditing is required to ensure that the files stored on the CSS are fully and intactly maintained. To reduce the user overhead, the TPA performs periodic integrity audits. To do this, the TPA first chooses a random challenge and it sends it to the CSS. The CSS responds by generating a corresponding proof using the stored file. Then the TPA verifies that the response is valid and completes the integrity audit.

1) FIRST UPLOAD PROCEDURE

Let us consider the case where a client U_{ID} uploads F to CSS. To do this, he should first generate a convergent key K and a file ID T_F. This process can be performed using existing schemes such as DupLESS [11]; this is omitted in this paper. The client sends the Upload Req message to the CSS including $\{U_{ID}, T_F\}$. The CSS checks if a duplicated file exists using T_F. If not exists, the CSS performs the First upload procedure by sending the First upload Req message to the client. Then, the client computes the ciphertext CT by encrypting F with the convergent key K. $CT = Enc_K(F)$, where $Enc_k()$ is an encryption mechanism such as AES in CTR mode [5] and k is a secret key. We assume that CT is divided into n blocks. The client also computes audit tags $\{t_i\}_{i=1}^n$ to periodically run the integrity audit procedure by the TPA. $t_i = \left(BLSHash(i) \cdot u^{CT_i}\right)^{\alpha}$, where u is a generator selected randomly by the client, and CT_i is the i-th ciphertext block. The client replies $\left\{CT, \{t_i\}_{i=1}^n\right\}$ to CSS, and then CSS stores $\left\{T_F, CT, \{t_i\}_{i=1}^n, U_{ID}\right\}$. In addition, the CSS issues a receipt including $Sig_{CSS}(T_F, U_{ID})$ to the client. The client sends the Audit Info message including $\{T_F, U_{ID}, u, n, Sig_{CSS}(T_F, U_{ID})\}$ to the TPA. The client keeps only (T_F, n) and deletes the other information. Fig. 2 shows the First upload procedure.

One consideration here is the management of the convergence key K. The convergence key K may be stored securely by each client individually. Alternatively, it can upload $ek = Enc_{mk}(K)$, which is encrypted using the secret master key mk of each client, to the CSS. In this case, the client only needs to keep mk secret. However, the CSS should store ek of each user for the duplicate file.

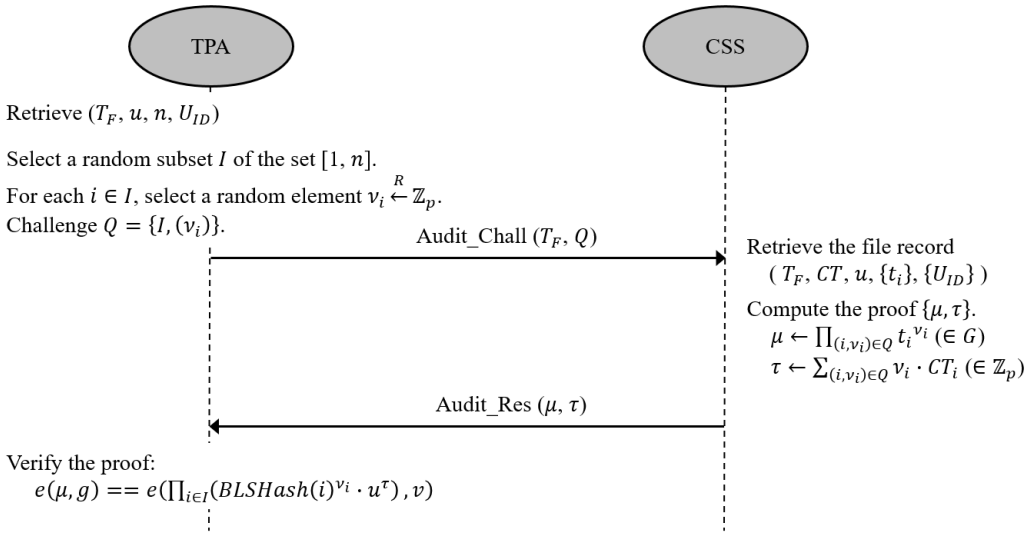


FIGURE 3. Integrity auditing procedure.

2) INTEGRITY AUDITING PROCEDURE

The TPA periodically checks the integrity of the data stored in the CSS. To do this, the TPA first selects a random subset $I \in [1, n]$, and then randomly selects v_i from $\mathbb{Z}_p$, for each $i \in I$. The challenge values for integrity auditing are $Q = \{I, (v_i)\}$. The TPA sends the Audit_Chall message with (T_F, Q) to the CSS. Then, the CSS computes the proof values $\{\mu, \tau\}$ corresponding to the challenge as follows;

$$\mu \leftarrow \prod_{(i,v_i)\in Q} t_i^{v_i} \quad (\in G) \tag{1}$$

and

$$\tau \leftarrow \sum_{(i,v_i)\in Q} v_i \cdot CT_i \quad (\in \mathbb{Z}_p). \tag{2}$$

The CSS replies to the Audit_Res message with $\{\mu, \tau\}$. The TPA verifies the proof as follows:

$$e(\mu, g) == e\left(\prod_{i\in I}\left(BLSHash(i)^{v_i} \cdot u^{\tau}\right), v\right). \tag{3}$$

Fig. 3 shows the Integrity auditing procedure.

3) SUBSEQUENT UPLOAD PROCEDURE

If the CSS has already stored the same file, the Subsequent upload procedure is performed for the deduplication (see Fig. 4). To verify the ownership of the client $U_{ID'}$, the CSS runs the PoW protocol. To do this, the CSS generates the challenge values $Q = \{I, (v_i)\}$, where I is a random subset of the set $[1, n]$ and v_i is a random element selected in $\mathbb{Z}_p$. The CSS sends the PoW Chall message with (u, Q) to the client. Then, the client derives the corresponding proof values $\{\mu, \tau\}$, computes $t_i' = \left(BLSHash(i) \cdot u^{CT_i}\right)^{\alpha'}$ for each $i \in I$, and then computes $\mu \leftarrow \prod_{(i,v_i)\in Q} t_i'^{v_i} \ (\in G)$, $\tau \leftarrow \sum_{(i,v_i)\in Q} v_i \cdot CT_i (\in \mathbb{Z}_p)$. The client sends the PoW Res message with the proof $\{\mu, \tau\}$ to the CSS. Finally, the CSS verifies the proof by checking that the following holds:

$$e(\mu, g) == e\left(\prod_{i\in I}\left(BLSHash(i)^{v_i} \cdot \mu^{\tau}\right), v'\right). \tag{4}$$

Upon successful completion of the PoW procedure, the CSS can issue a receipt to the client by adding a user ID, $U_{ID'}$ and a reference for the file to the list of corresponding file access rights in order to confirm the file ownership. The client may give the receipt to the TPA. Then, the TPA may examine the file's integrity using the receipt. If the CSS manages encrypted convergent keys, the client must upload $ek' = Enc_{mk'}(K)$, encrypted using its secret master key mk', to the CSS.

IV. ANALYSIS

The proposed scheme satisfies the security objectives mentioned above. From the privacy perspective, the proposed method outsources the encrypted ciphertext to the CSS using the convergent encryption key. In the integrity auditing process, the TPA can also partially obtain the information about the ciphertext. Assuming that the convergent key generation process is performed through the OPRF (oblivious pseudo random function) protocol with the trusted key server as in DupLESS [11], it provides security against offline brute-force attacks. An attacker who does not obtain the convergent key K cannot get any information from the outsourced ciphertext, except the information of duplication. The proposed scheme also supports secure deduplication by providing deduplication for the ciphertext and performing the PoW protocol. This also depends on the security of the convergent key, as mentioned above, and the security of BLS signature based HLA [14].

The TPA then audits the integrity of the data without user intervention. It depends on the security of the BLS-based

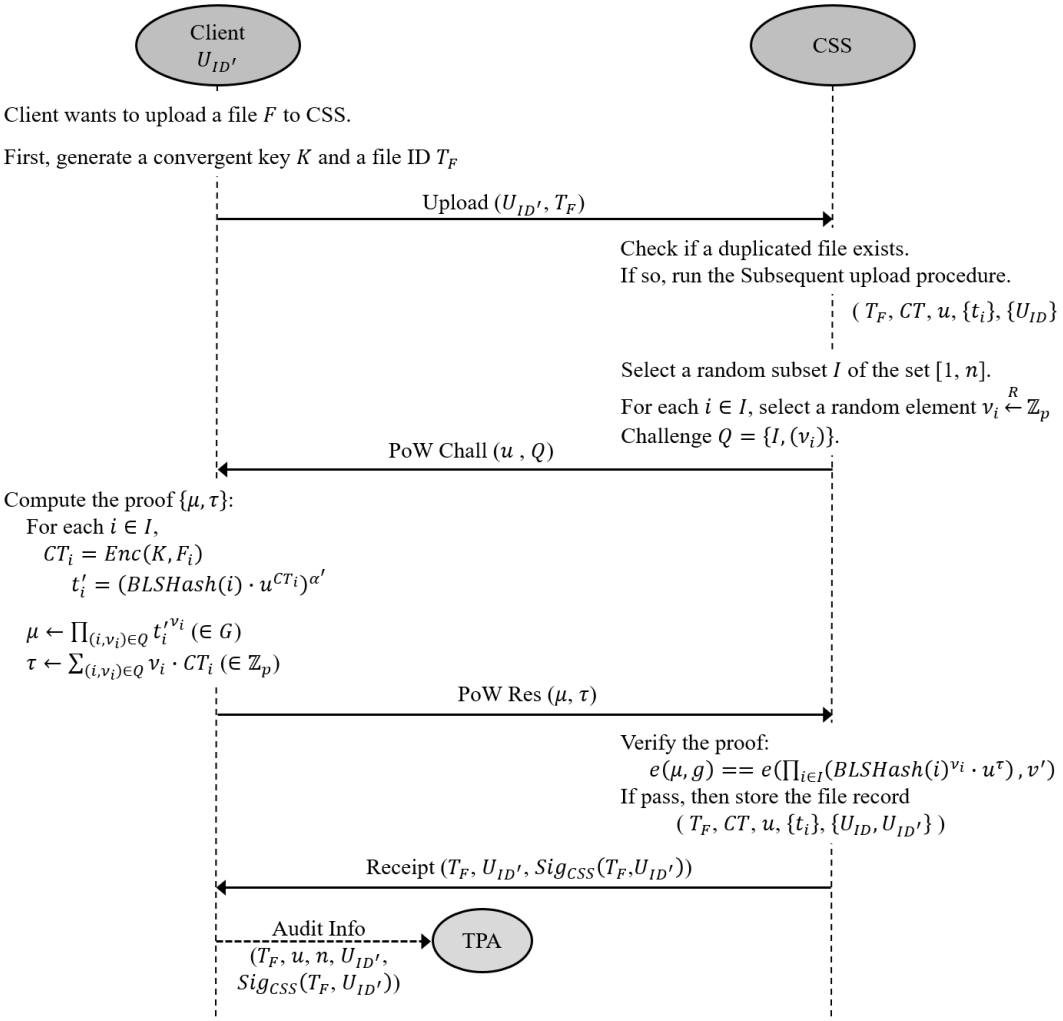


FIGURE 4. Subsequent upload procedure with PoW.

homomorphic authentication tag, and its security has been analyzed in [14]. Thus, the proposed scheme supports public verifiability. Finally, if the CSS keeps the data intact, it can pass the verification by the TPA. This is also provided by the security of the BLS-based homomorphic authentication tag used, thus ensuring the correctness of the storage.

In order to provide a comparison with the existing schemes, first, the scheme proposed in [18] is considered. This scheme supports integrity auditing and deduplication using a polynomial-based authentication tag and a homomorphic linear tag. During the setup process, the user computes a homomorphic linear tag and uploads it to the cloud server. Then, the TPA performs integrity auditing with the cloud server through the interaction using a polynomial-based authentication tag. In the deduplication process, when the cloud server randomly selects a set of block indexes for the PoW, the server sends them to the user. Then, the user transmits the corresponding plaintext blocks as the response. Then, the cloud server verifies the file ownership by verifying the validity of the received blocks using the pairing operation. The biggest problem with this scheme is that the data is used as a plain text on the cloud side; therefore, it does not support secure deduplication. In addition, regardless of the file duplication, users always have to compute authentication tags. This results in a high computational overhead.

In the scheme proposed in [12], the client uploads a file to a TPA that is assumed to be honest. This is a very strong assumption, since most of TPAs in the existing public auditing-related papers are assumed to be semi-honest. It also wastes the bandwidth on the client side, because it always transmits the file to the TPA. When the client uploads the file to the TPA, the TPA computes a homomorphic signature for integrity auditing and uploads it to the cloud server along with the file.

The proposed scheme improves the problems of the above two methods. Table 1 shows the comparison with the existing schemes. The TPA of the proposed scheme is assumed to be semi-honest, and also does not upload the file to the TPA. In addition, the proposed method supports deduplication and integrity auditing for the encrypted data, and the client only

TABLE 1. Comparison with existing schemes.

	Secure deduplication	Privacy-preserving public auditing	Efficiency
[18]	X (because of available in its plain form on CSS)	O	Require two different authentication tags for PoW and for auditing, respectively, regardless of duplication.
[12]	O	X (because the file is uploaded to TPA)	Loss of bandwidth advantages in the client side
Proposed scheme	O	O	Require a computation of only one tag of both authentication tags, and ensure bandwidth advantages in the client side

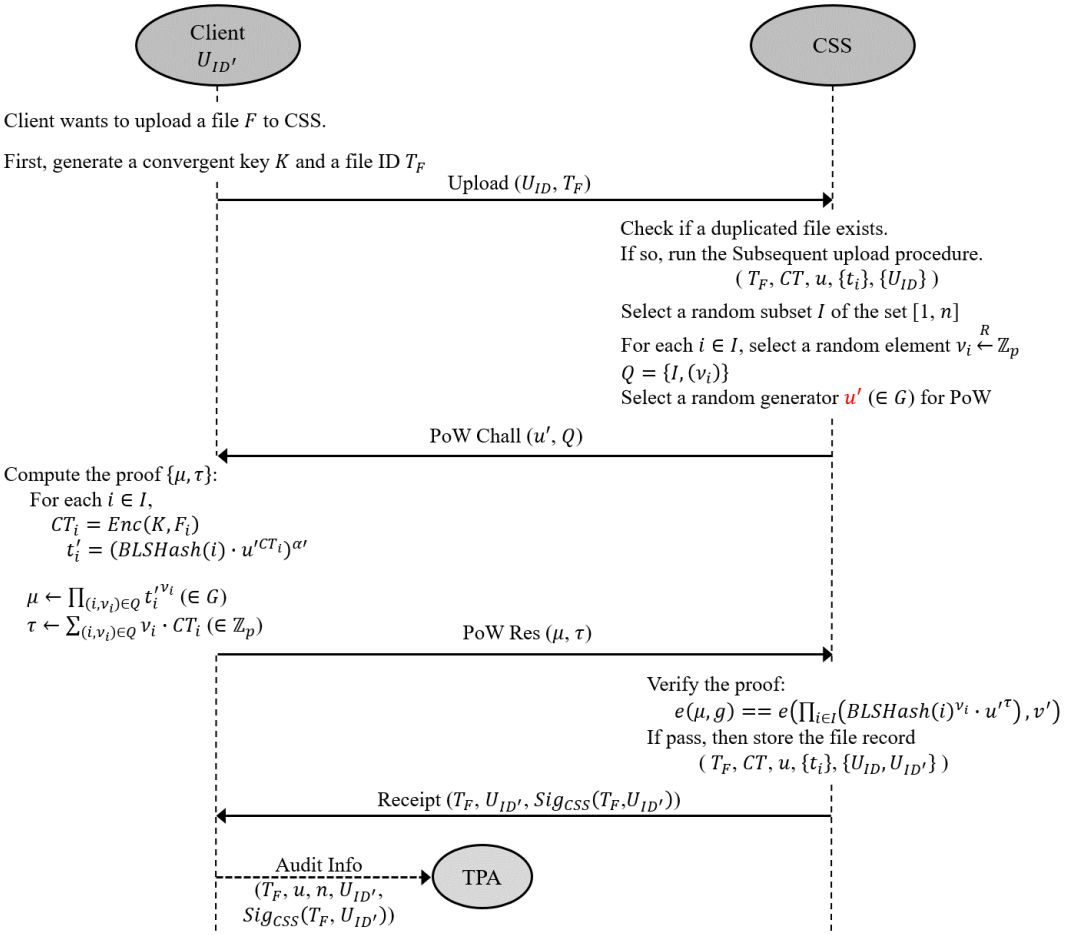


FIGURE 5. Subsequent upload process with improved security.

needs to perform a single authentication tag computation. It has similar computational overhead in each case of the first upload and duplicate upload. That is, in the case of the first upload, the authentication tag for the integrity audit is computed, and in the case of deduplication, the authentication tag for the PoW is generated. Therefore, it provides better efficiency than the existing schemes in the viewpoint of client-side computational overhead.

V. VARIANCES

In Section III, we designed a new secure deduplication supporting public auditing, and proved its security and efficiency in Section IV. The proposed scheme is secure under a reasonable security model and its performance is better than the existing schemes as shown in Section IV. Here, we provide some techniques to achieve greater security and better performance.

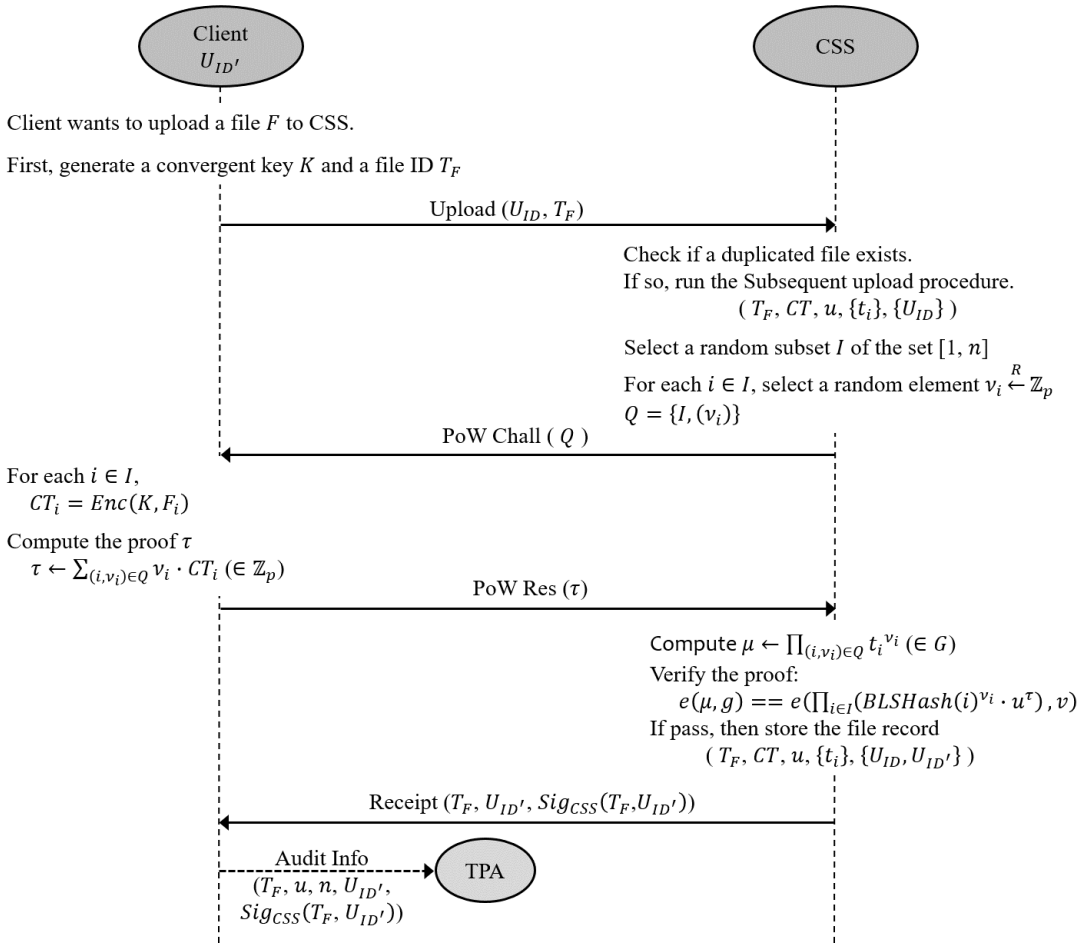


FIGURE 6. Subsequent upload process for improving the computational overhead on the client side.

A. IMPROVEMENT FROM THE VIEWPOINT OF SECURITY

In this section, we will consider a slightly stronger attack scenario, which was not considered in Section IV. Recall that we assumed the original data holder to be a reliable entity who behaves honestly. Hence, we used the assumption to analyze the proposed scheme. However, in this section, we discuss a possible attack scenario, which can be performed by the valid user who is a legitimate data holder, and provide a countermeasure for the attack by slightly modifying the scheme in Section III.

Recall that the proposed model described in Section III uses the same generator u to generate the authentication tag for integrity verification and the authentication tag for PoW. In the case of the duplicated upload, it may be possible to attack the server in PoW process by acquiring the authentication tag of the already uploaded file and using it to pass the ownership proof procedure. When the adversary is an outsider who cannot easily obtain the tag, it is not easy to mount the attack. However, if the original data holder helps an adversary to obtain the authentication tag, it is easy to mount the attack. As recognized in [8], the original file holder can utilize the storage service as a content distribution network, and this could be a threat to storage services. Hence, if we consider such a stronger adversary, we need a countermeasure to prevent a legitimate user from helping others to obtain the stored data.

Note that it is not possible to prevent the data holder from giving his data to others, and therefore the goal of the countermeasure is to make it difficult for an adversarial user to use the storage service as a content distribution network. As a concrete countermeasure to alleviate this threat, we can consider a random selection of a generator for each PoW process. The CSS randomly chooses a generator u' and sends it to the client as the PoW Chall message. Then, it is possible for the CSS to perform more secure PoW by verifying whether an appropriate response is returned by the client. Even though the original file holder can still help other users to obtain the ownership, the user should have the entire file and perform costly operations to do so. Hence, in the data holder's viewpoint, it is better to give the file to the adversary instead of helping the adversary to pass the PoW procedure. Fig. 5 shows the Subsequent upload process with improved security.

In the case of a duplicate file upload, the CSS performs the PoW process. To do this, the CSS chooses a random generator u' and challenge values $Q = \{I, (v_i)\}$, where the challenge Q

is generated as in Section III. The CSS sends the PoW Chall message with (u', Q) to the client. Then, the client returns the corresponding proof values (μ, τ) computed by using the generator u' and its own public key α'. The CSS verifies the proof with u' and the client's public key v' by checking whether the following holds;

$$e(\mu, g) == e\left(\prod_{i \in I} \left(BLSHash(i)^{v_i} \cdot \mu'^{\tau}\right), v'\right). \quad (5)$$

After obtaining the ownership, the client and the CSS use the same generator u to perform the integrity auditing. However, it does not a matter anymore since the client cannot help others by using the values to pass the PoW to obtain the ownership of the file F.

B. IMPROVEMENT FROM THE VIEWPOINT OF EFFICIENCY

At present, a variety of devices are used to generate and use data in storage services. Though the capability of the devices has improved more than ever before, we still need to design light schemes for storage services due to the increase in size of data. In this viewpoint, we design a technique that can permit a client to pass some costly operations to the CSS in the upload procedure.

To reduce the computational complexity, as shown in Fig. 6, the process of uploading duplicate files can be modified. The client uploading the duplicate file computes only the authentication tag τ for each CT_i, unlike in Section III. That is, the client does not compute μ and sends the PoW Res message with only the proof τ to the CSS. Then, the CSS computes μ and verifies τ.

This reduces the amount of computation on the client side, while the CSS's computational overhead increases relatively. However, when the client is a lightweight device such as a mobile device it is advantageous to transfer a part of the computation to the CSS, which has a a higher performance than the client. Moreover, we can implement the online step by choosing $Q = \{I, (v_i)\}$ and pre-computing μ before a subsequent upload process is initiated by a new client. If we apply the pre-computation technique, we can reduce the computational complexity without increasing the cost for the CSS in the online step.

VI. CONCLUSION

When storing data on remote cloud storages, users want to be assured that their outsourced data are maintained accurately in the remote storage without being corrupted. In addition, cloud servers want to use their storage more efficiently. To satisfy both the requirements, we proposed a scheme to achieve both secure deduplication and integrity auditing in a cloud environment. To prevent leakage of important information about user data, the proposed scheme supports a client-side deduplication of encrypted data, while simultaneously supporting public auditing of encrypted data. We used BLS signature based homomorphic linear authenticator to compute authentication tags for the PoW and integrity auditing. The proposed scheme satisfied the security objectives, and improved the problems of the existing schemes. In addition, it provides better efficiency than the existing schemes in the viewpoint of client-side computational overhead. Finally, we designed two variations for higher security and better performance. The first variance guarantees higher security in the sense that a legitimate user can be an adversary. The second variance provides better performance from the perspective of the clients, by permitting low-powered clients to perform upload procedure very efficiently by passing on their costly operations to the CSS.

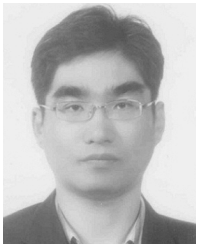
REFERENCES

[1] G. Ateniese *et al.*, "Provable data possession at untrusted stores," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, Alexandria, VA, USA, 2007, pp. 598–609.
[2] G. Ateniese, R. Di Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in *Proc. 4th Int. Conf. Secur. Privacy Commun. Netowrks*, Istanbul, Turkey, 2008, Art. no. 9.
[3] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," *J. Cryptol.*, vol. 17, no. 4, pp. 297–319, 2004.
[4] Y. Dodis, S. Vadhan, and D. Wichs, "Proofs of retrievability via hardness amplification," in *Proc. 6th Theory Cryptogr. Conf.*, San Francisco, CA, USA, 2009, pp. 109–127.
[5] M. Dworkin, "Recommendation for block cipher modes of operation: Methods and techniques," NIST, Gaithersburg, MD, USA, Tech. Rep. SP-800-38A, 2001.
[6] C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *Proc. 16th ACM Conf. Comput. Commun. Secur.* Chicago, IL, USA, 2009, pp. 213–222.
[7] J. Gantz and D. Reinsel, "The digital universe decade—Are you ready?" International Data Corporation, China Oceanwide, MA, USA, White Paper IDC-925, 2010.
[8] S. Halevi, D. Harnik, B. Pinkas, and A. Shulman-Peleg, "Proofs of ownership in remote storage systems," in *Proc. 18th ACM Conf. Comput. Commun. Secur.* Chicago, IL, USA, 2011, pp. 491–500.
[9] D. Harnik, B. Pinkas, and A. Shulman-Peleg, "Side channels in cloud services: Deduplication in cloud storage," *IEEE Security Privacy*, vol. 8, no. 6, pp. 40–47, Nov./Dec. 2010.
[10] A. Juels and B. S. Kaliski, Jr., "Pors: Proofs of retrievability for large files," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, Alexandria, VA, USA, 2007, pp. 584–597.
[11] S. Keelveedhi, M. Bellare, and T. Ristenpart, "DupLESS: Server-aided encryption for deduplicated storage," in *Proc. 22nd USENIX Secur. Symp.*, Washington, DC, USA, 2013, pp. 179–194.
[12] J. Li, J. Li, D. Xie, and Z. Cai, "Secure auditing and deduplicating data in cloud," *IEEE Trans. Comput.*, vol. 65, no. 8, pp. 2386–2396, Aug. 2016.
[13] X. Liu, W. Sun, H. Quan, W. Lou, Y. Zhang, and H. Li, "Publicly verifiable inner product evaluation over outsourced data streams under multiple keys," *IEEE Trans. Services Comput.*, vol. 10, no. 5, pp. 826–838, Sep./Oct. 2017.
[14] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, Melbourne, VIC, Australia, 2008, pp. 90–107.
[15] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 5, pp. 847–859, May 2011.
[16] T. Y. Youn, K. Y. Chang, K. R. Rhee, and S. U. Shin, "Public audit and secure deduplication in cloud storage using BLS signature," *Res. Briefs Inf. Commun. Technol. Evol.*, vol. 3, pp. 1–10, Nov. 2017, Art. no. 14.
[17] J. Yuan and S. Yu, "Proofs of retrievability with public verifiability and constant communication cost in cloud," in *Proc. Int. Workshop Secur. Cloud Comput.* Hangzhou, China, 2013, pp. 19–26.
[18] J. Yuan and S. Yu, "Secure and constant cost public cloud storage auditing with deduplication," in *Proc. IEEE Conf. Commun. Netw. Secur. (CNS)* National Harbor, MD, USA, Oct. 2013, pp. 145–153.

TAEK-YOUNG YOUN received the B.S., M.S., and Ph.D. degrees from Korea University in 2003, 2005, and 2009, respectively. Since 2016, he has been serving as an Associate Professor with the University of Science and Technology, Daejeon, South Korea. He is currently a Senior Researcher with the Electronics and Telecommunications Research Institute, Daejeon. His research interests include cryptography, information security, authentication, data privacy, and security issues in various communications.

KU-YOUNG CHANG received the B.S., M.S., and Ph.D. degrees in mathematics from Korea University, Seoul, South Korea, in 1995, 1997, and 2000, respectively. He is currently a Principal Researcher with the Cryptography Research Section, Electronics and Telecommunication Research Institute, Daejeon, South Korea. His research interests include cryptography, data privacy, and finite field theory.

KYUNG-HYUNE RHEE received the M.S. and Ph.D. degrees from the Korea Advanced Institute of Science and Technology, Daejeon, South Korea, in 1985 and 1992, respectively. He was a Senior Researcher with the Electronic and Telecommunications Research Institute, Daejeon, from 1985 to 1993. He was also a Visiting Scholar with The University of Adelaide, Australia, The University of Tokyo, Japan, the University of California at Irvine, USA, and Kyushu University, Japan. He has served as the Chairman of the Division of Information and Communication Technology, Colombo Plan Staff College for Technician Education, Manila, Philippines. He is currently a Professor with the Department of IT Convergence and Application Engineering, Pukyong National University, Busan, South Korea. His research interests center on multimedia security and analysis, key management protocols, and mobile ad hoc and VANET communication security.



SANG UK SHIN received the M.S. and Ph.D. degrees from Pukyong National University, Busan, South Korea, in 1997 and 2000, respectively. He was a Senior Researcher with the Electronics and Telecommunications Research Institute, Daejeon, South Korea, from 2000 to 2003. He is currently a Professor with the Department of IT Convergence and Application Engineering, Pukyong National University. His research interests include digital forensics, e-Discovery, cryptographic protocol, mobile and wireless network security, and multimedia content security.