Transparent Provable Data Possession Scheme for Cloud Storage

OGISO, Shinobu

MOHRI, Masami

SHIRAISHI, Yoshiki

Transparent Provable Data Possession Scheme for Cloud Storage

Shinobu OGISO
Gifu University
Gifu 501-1193, Japan
y4525022@edu.gifu-u.ac.jp

Masami MOHRI
Gifu University
Gifu 501-1193, Japan
mmohri@gifu-u.ac.jp

Yoshiki SHIRAISHI
Kobe University
Hyogo 657-8501, Japan
zenmei@port.kobe-u.ac.jp

Abstract- Provable Data Possession (PDP) is one of the data security techniques to make sure that the data stored in the cloud storage exists. In PDP, the integrity of the data stored in the cloud storage is probabilistically verified by the user or a third-party auditor. In the conventional PDP, the user creates the metadata used for audition. From the viewpoint of user convenience, it is desirable to be able to audit without operations other than uploading. In other words, the challenge is to provide a transparent PDP that verifies the integrity of files according to the general cloud storage system model so as not to add operations to users. We propose a scheme in which the cloud generates the metadata used during verification, and the user only uploads files. It is shown that the proposed scheme is resistant to the forgery of cloud proof and the acquisition of data by a third-party auditor.

Keywords- cloud storage, audit, data integrity, provable data possession, discrete logarithm problem

I. INTRODUCTION

Cloud storage systems are used in many areas for data sharing, data storage, and data management. Several problems have been pointed out regarding cloud storage systems. One is that files and data stored in cloud storage may not be consistent. For example, due to misoperation or lack of management by the cloud administrator [1][2][3], the integrity may be partially or completely losing the data stored in the cloud. Therefore, to achieve a more secure cloud storage, it is expected that the users can check whether the integrity of data stored by them is preserved.

The simple idea is to return the data stored in the cloud to the user once, and the user can check the data integrity one by one. However, this scheme requires longer verification time and higher cost as the amount of stored data increases. Therefore, the auditing system called Provable Data Possession (PDP) was proposed by Ateniese et al. [4]. PDP is a scheme in which the stored data is divided into multiple blocks, and a user or a third-party auditor verifies the file integrity for blocks randomly selected from the blocks. Probabilistically verifying data integrity results in lower computational cost than the simple scheme. Various PDPs have been proposed based on the scheme of Ref. [4].

In Refs.[4],[7]-[19], the user generates the metadata used when auditing by the user or the auditor. However, when using a cloud storage system in the real world, users often only upload data to the cloud [5][6]. In other words, the user does not often perform additional processing such as encryption on the file, and many storages will encrypt and store the file on the cloud.

We aim to realize PDP that allows the user to audit only by uploading the file to the cloud. The challenge is to provide a new PDP under the system model that the metadata used for auditing is generated by cloud storage and can be verified by a third-party for integrity of the file. This paper proposes a discrete logarithm problem-based PDP in the system model, which has the following security properties: (1) When the data stored in the cloud is lost due to management mistakes, it is difficult for the cloud to forge the proof. (2) A malicious auditor cannot obtain information about the stored files from the data used during integrity verification.

II. RELATED WORK

PDP (Provable Data Possession)[4] is a probabilistic file possession verification originally proposed by Ateniese et al. The system flow is as follows; First, the user creates metadata to be used for auditing when storing the file in the cloud. It stores the metadata and the file in the cloud , and deletes the file that is local to itself. Then, when verifying possession of a file, not the entire file is to be verified, but some of the blocks obtained by dividing the file are selected, and verification is performed using the selected blocks and the metadata according to selected blocks. There are two types of the above audit: one is performed by the user and another is performed by a third-party auditor. PDP provides a function that verifies whether there is data in cloud storage. In other words, we can verify whether the integrity of the files in the cloud storage is maintained by using PDP.

Ateniese et al. firstly proposed a PDP that made the efficiency of [4] even higher [7]. The scheme of [4] uses RSA encryption, but the scheme of [7] becomes possible to verify efficiently by using common key encryption. Kaaniche et al. [8] proposed a zero-knowledge proof-based PDP that against data leakage. Wang et al. [9] proposed a PDP that enables batch auditing that allows an auditor to simultaneously perform audits requested by multiple users, and Wang [10] proposed an auditing scheme when files are divided and saved on multiple servers. In [10], in addition to users, auditors, and the cloud, an entity called Combiner that divides and integrates the data sent from each entity appears. A third-party auditing scheme using ZSS short signature by a bilinear map is proposed [11]. Luo et al.

978-1-7281-5628-6/20/$31.00 ©2020 IEEE

proposed an integrity verification system using BLS signatures and improved the performance of privacy protection for a third-party auditor [12]. Wang et al. proposes an integrity verification system for data that has been transferred between the acquired company and the acquiring company [13].

The generation and management of public/private key pairs used in PDP is difficult to manage in a system where PKI (public key infrastructure) is performed as the number of users increases, and verification may not be able to be performed smoothly. A scheme has been proposed that reduces the burden of public key management by using ID-based encryption for PDP [14]. A PDP called CLPDP (certificateless provable data possession) was also proposed to solve the key escrow problem, which is a problem in ID-based encryption [15]. Ming et al. [16] proposed a CLPDP with improved security and performance.

A PDP that supports dynamic processing such as updating and erasing files in cloud storage has also been proposed. Such the techniques as a rank-based authenticated skip list [17] and Merkel Hash tree [18] are used for file saving and dynamic processing in Dynamic PDP (DPDP). Barsoum et al. proposed audit systems named MB-DMCPDP and TB-DMCPDP, that use skip list and MHT for dynamic processing of multiple copied files stored on the server [19].

In the above existing PDP, a user generates a metadata used for the audit by the user and the auditor. However, in the general case of using cloud storage in the real world, many users only upload files to the cloud without encryption, and the cloud side often performs additional processing for data security. In other words, the existing PDP system model has a part that is different from the operation of the cloud storage system currently used in the world. Considering the case where PDP is actually used in the real world, although the conventional PDPs contribute data security as important techniques, the users are required to additional operation. If the cloud can generate the metadata, the user will not be burdened and the existing cloud system can be maintained while auditing by PDP. Therefore, the challenge is to provide a transparent PDP that verifies the integrity of files according to the general cloud storage system model so as not to add operations to users. We propose a system in which the user only uploads files and the audit is performed by a third-party. The scheme will be possible to perform audits while almost keeping the specifications of the existing cloud storage system from the view of user.

III. Proposed Scheme

A. Definition of Algorithms

The proposed scheme consists of four polynomial time algorithms (*KeyGen, TagGen, GenProof, ProofCheck*) such that:

$KeyGen(1^k) \rightarrow pk$ is run by the user to generate public information $pk = (p, q, g)$.

$TagGen(pk, M) \rightarrow (t, T, \Sigma = \{T_{m_1}, T_{m_2}, \dots, T_{m_n}\})$ is run by the cloud to generate the verification metadata. It takes as inputs the public information (p, q, g), and a file M, and returns the verification metadata $(t, T, \Sigma = \{T_{m_1}, T_{m_2}, \dots, T_{m_n}\})$.

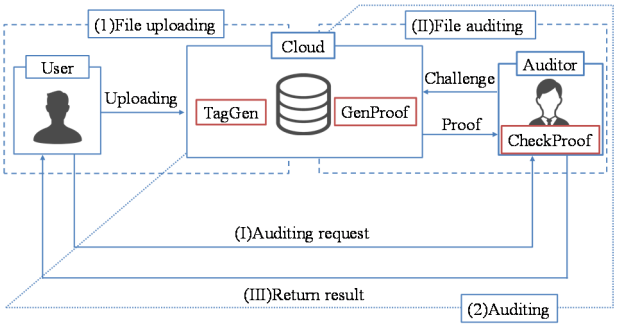

Fig.1 System model of proposed scheme

$GenProof(pk, chal, M, t, \Sigma) \rightarrow \gamma$ is run by the cloud to generate a proof of possession. It takes as inputs public information (p, q, g), a challenge $chal$, file M, metadata t and an ordered collection $\Sigma = \{T_{m_1}, T_{m_2}, \dots, T_{m_n}\}$ which is the verification metadata corresponding to the blocks in M .It returns a proof of verification γ for the blocks in file M that are determined by the $chal$.

$CheckProof(pk, chal, \gamma, T) \rightarrow \{success, failure\}$ is run by the auditor to validate a proof of possession. It takes as inputs public information pk, a challenge $chal$, metadata T and a proof of verification γ. It returns a result of whether γ is a correct proof of possession for blocks determined by $chal$.

B. System Model

The proposed scheme is built on the system model in Fig.1. There are three entities (the user, the cloud storage and, the auditor) in this system model.

The system model can be constructed in two phases, **File uploading** and **Auditing**.

(1) **File uploading:** The user runs $KeyGen(1^k) \rightarrow pk$ and uploads the file M in the cloud storage. The cloud runs $TagGen(pk, M) \rightarrow (t, T, \Sigma = \{T_{m_1}, T_{m_2}, \dots, T_{m_n}\})$ and then stores (M, t, Σ) and sends T to the user.

(2) **Auditing:**

I. The user requests the cloud to audit the file M and sends the metadata T to the auditor.

II. The auditor sends the challenge $chal$ to the cloud. The cloud runs $GenProof(pk, chal, M, t, \Sigma) \rightarrow \gamma$ and sends to the auditor the proof of possession γ.

III. Finally, the auditor runs $CheckProof(pk, chal, \gamma, T)$ to validate a proof of possession and then sends a verification result to the user.

C. Security Model

We define the following three security properties based on Refs. [4] and [20].

(1) **Correctness.** If both the cloud and the auditor are honest, the cloud can pass the verification. In other words, the verification equation used by the auditor holds correctly when the correct proof against the auditor's challenge is given by the cloud.

(2) **Integrity Protection.** The integrity of the files stored in the cloud may be lost because of mismanagement. In that case, the cloud must not be able to generate a valid proof which the auditing is valid. This property indicates that

probabilistic possession proof cannot be performed if the cloud does not store the file block used for auditing.

(3) **Privacy Protection.** When a malicious auditor conducts an audit, it may be possible to try to obtain some information from the file to be audited. This property indicates that malicious auditors should not be able to obtain any information from the files they audit.

D. Proposed Transparent PDP Scheme

It is assumed that a large prime p and the element g of Z_p^* having a large prime order q are disclosed.

File uploading phase

The file uploading phase consists of the following steps from (1) to (3) as shown in Fig. 2.

(1) The cloud divides file M into n blocks such as $M = \{m_1, m_2, \dots, m_n\}$.

(2) The cloud computes metadata as follows:

$$t_{m_i} = h(m_i)\ mod\ q \ \rightarrow t = \sum_{i=1}^{n} t_i\ mod\ q$$
$$T_{m_i} = g^{-h(m_i)}\ mod\ p\ (i = 1, \dots, n)$$
$$T = g^{-t}\ mod\ p$$

(3) The cloud stores $(t, M, \{T_{m_1}, T_{m_2}, \dots, T_{m_n}\})$, and sends a metadata T to the user.

Auditing phase

The file auditing is shown from I to III in Fig.1.

I. To start auditing a file, a user sends a metadata T to an auditor.

II. The file audit consists of the following steps from (1) to (7) shown in Fig. 3.

(1) The auditor sends a challenge $chal = (k, c_1)$ to the cloud, where k and c_1 are a seed of a pseudo-random permutation π and a random number to use to generate a proof of possession, respectively.

(2) The cloud selects c_2 and computes the indices of the blocks for which is not used for auditing using a pseudo-random permutation (PRP) π .

$$\pi_k(j) = i_j (1 \le i_j \le n, 1 \le j \le c_2)$$

(3) The cloud computes a metadata set $(\bar{t}, \bar{T})$ as follows:

$$\bar{t}_{i_j} = h\left(m_{i_j}\right)\ mod\ q \ \rightarrow \bar{t} = \sum h(m_{i_j})\ mod\ q$$
$$\bar{T} = T_{m_{i_1}} \cdot \dots \cdot T_{m_{i_{c_2}}} mod\ p$$

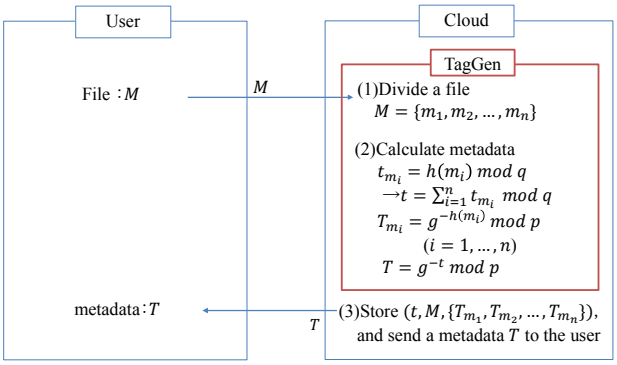

Fig.2 File uploading phase

(4) The cloud calculates x and y in the following equations.

$$x = g^r\ mod\ p$$
$$y = r + (t - \bar{t})c_1\ mod\ q$$

(5) The cloud returns $(x, y, \bar{T})$ to the auditor as a proof for $chal$.

(6) The auditor receives $(x, y, \bar{T})$ and calculates

$$T' = T * \bar{T}$$

from T and $\bar{T}$.

(7) The auditor checks whether $x = g^y (T')^{c_1}$ holds. If this equation holds, the auditor can verify that file integrity is preserved.

III. The auditor replies to the result of step (7) to the user for the file M.

IV. Security

We give some security considerations on the security properties.

(1) **Correctness**: The correctness of the proposed scheme can be proven as follows.

Proof :

$$x = g^y (T')^{c_1}$$
$$= g^y (T * \bar{T})^{c_1}$$
$$= g^{r+(t-\bar{t})c_1} (g^{-t} * g^{\bar{t}})^{c_1}$$

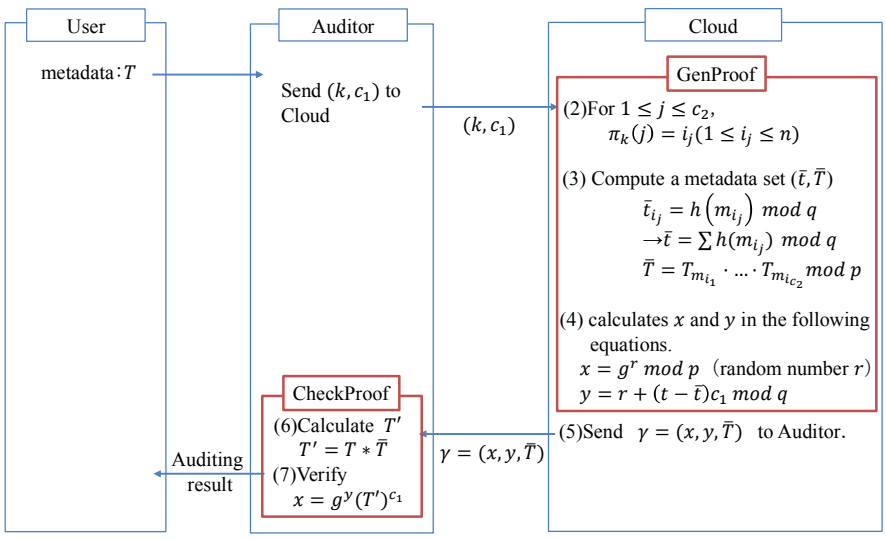

Fig.3 Auditing phase

$= g^{r+(t-\bar{t})c_1}(g^{-(t-\bar{t})})^{c_1}$
$= g^{r+(t-\bar{t})c_1}(g^{-(t-\bar{t})c_1})$
$= g^r = x$

(2) **Integrity Protection:** Suppose that the file stored in the cloud is missing. The cloud tries to forge a proof to pass the audit of the file. There are the file $M = \{m_1, m_2, \dots, m_n\}$ and the metadata $(t = \sum_{i=1}^{n} h(m_i) \ mod \ q, , \Sigma = \{T_{m_1}, T_{m_2}, \dots, T_{m_n}\})$ in the cloud storage. Let us consider whether it is possible to forge a proof for a missing file block $m_k (1 \leq k \leq n)$.

First, the cloud tries to get $h(m_k)$ from $T_{m_k} = g^{-h(m_k)} \ mod \ p$. Because of the discrete logarithm problem, the cloud cannot get $h(m_k)$ from T_{m_k}. Since the metadata t is the sum of the hash values of all file blocks, the hash values of file blocks other than m_k are also required to obtain $h(m_k)$. However, since the file blocks used for auditing are determined by a pseudo-random permutation π , the hash values of all file blocks cannot be calculated. Moreover, since the file block m_k is missing, it is not known what value the complete file block m_k indicates. Therefore, it is obvious that the correct hash value $h(m_k)$ cannot be calculated. From the above, it is not possible to forge a proof for a missing file block.

(3) **Privacy Protection:** We consider whether a malicious auditor can get some information about a file from the file being audited. In the audit procedure, the data that the auditor can obtain is $(T, x, y, \bar{T})$. We will check whether the data related to the file to be audited can be obtained from the data.

First, consider the two values $T = g^{-t} \ mod \ p$ and $\bar{T} = g^{\bar{t}} \ mod \ p$. Due to the discrete logarithm problem, the hash value $(t, \bar{t})$ of the file block cannot be obtained from these values. Next, consider whether $t - \bar{t}$ can be obtained from y and c_1 . However, because of the discrete logarithm problem, the value of the random number r cannot be obtained from $x = g^r \ mod \ p$. Since the random number r is unknown, $t - \bar{t}$ cannot be obtained from y. From the above, the auditor cannot obtain information about the file at the time of audit.

V. Conclusion

In this paper, we proposed a transparent PDP scheme in which the cloud generates the metadata for verification, and the user only uploads files. This scheme only requires the user to upload the file, keeps the cloud from forging a proof of possession and prevents the malicious auditor from getting the information of the file. With the proposed scheme, there is no way to confirm whether the verification result sent from the auditor is correct. Formal security proof will be given as a future work.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number JP18K04133, JP19K11963.

References

[1] T. Akervik, "WHAT ARE THE CHANCES OF LOSING INFORMATION IN CLOUD STORAGE?," marconet, Feb 8, 2019 , https://www.marconet.com/blog/what-are-the-chances-of-losing-information-in-cloud-storage .

[2] J. Rundle, "Human Error Often the Culprit in Cloud Data Breaches," The Wall Street Journal, Aug.27, 2019, https://www.wsj.com/articles/human-error-often-the-culprit-in-cloud-data-breaches-11566898203 .

[3] S. Nichols, "AWS celebrates Labor Day weekend by roasting customer data in US-East-1 BBQ," The Register, Sep. 2019, https://www.theregister.com/2019/09/04/aws_power_outage_data_loss/

[4] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, D. Song, "Provable data possession at untrusted stores," Proc. of the 14th ACM Conference on Computer and Communications Security (CCS '07), pp.598-609, 2007.

[5] sync.com, https://www.sync.com/

[6] tresorit, https://tresorit.com/home

[7] G. Ateniese, R.D. Pietro, L.V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," Proc. of the 4th International Conference on Security and privacy in communication networks (SecureComm'08), Article 9, pp.1-10, 2008.

[8] N. Kaaniche, E.E. Moustaine, M. Laurent, "A novel Zero-Knowledge scheme for proof of data possession in cloud storage applications," Proc. of 2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pp. 522-531, 2014.

[9] C. Wang, S.S.M. Chow, Q. Wang, K. Ren, W. Lou. "Privacy-preserving public auditing for secure cloud storage," IEEE Transactions on Computer, vol.62, no.2, pp.362-375, 2009.

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

[11] H. Zhu, Y. Yuan, Y. Chen, Y. Zha, W. Xi, B. Jia, Y. Xin, "A Secure and Efficient Data Integrity Verification Scheme for Cloud-IoT Based on Short Signature," IEEE Access, vol.7, pp.90036-90044, 2019.

[12] X. Luo, Z. Zhou, L. Zhong, J. Mao, C. Chen, "An effective integrity verification scheme of cloud data based on BLS signature," Security and Communication Networks, vol.2018, Article ID 2615249, 11 pages, 2018.

[13] H. Wang, D. He, A. Fu, Q. Li, Q. Wang, "Provable data possession with outsourced data transfer," IEEE Transactions on Services Computing, doi: 10.1109/TSC.2019.2892095.

[14] Y. Yu, M.H. Au, G. Ateniese, X. Huang, W. Susilo, "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, 2017.

[15] Y. Liao, Y. Liang, A. W. Oyewole, X. Nie, "Security analysis of a certificateless provable data possession scheme in cloud," IEEE Access, vol.7, pp.93259-93263, 2019

[16] Y. Ming, W. Shi, "Efficient Privacy-preserving certificateless provable data possession Scheme for cloud storage," IEEE Access, vol.7, pp.122091-122105, 2019.

[17] C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," Proc.

of the 16th ACM conference on Computer and Communications Security (CCS '09), pp. 213-222, 2009.

[18] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," Proc. of the 14th European Conference on Research in Computer Security (ESORICS'09), LNCS 5789, pp.355-370, 2009.

[19] A.F. Barsoum, M.A. Hasan, "On verifying dynamic multiple data copies over cloud servers," IACR eprint report 447, 2011. Available at http://eprint.iacr.org/2011/447.pdf

[20] M. Bellare, A. Palacio, "GQ and Schnorr identication schemes: proofs of security against impersonation under active and concurrent attacks," Advances in Cryptology - CRYPTO 2002, LNCS 2442, pp.162-177, 2002.