2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC) | 978-1-7281-5244-8/20/$31.00 ©2020 IEEE | DOI: 10.1109/ITAIC49862.2020.9339001

A Data Integrity Verification Scheme of Deduplication for Cloud Ciphertexts

Qingyun Hu
Department of Network Security Teaching and Research, Gansu Police Vocational College, Lanzhou, China
h_q_y_2007@163.com

***Abstract*—Cloud data deduplication can save cloud storage resources and network communication bandwidth, and improve cloud storage efficiency. A series of deduplication technologies are proposed to achieve cloud data deduplication. However, the traditional deduplication scheme mainly implements data deduplication without providing data integrity verification, so it cannot be verified whether the CSP correctly stores user data. This paper proposes a data integrity verification scheme of deduplication for cloud ciphertexts, including cloud ciphertext deduplication and cloud data integrity verification. In the data integrity verification, the proxy re-signature method is adopted. In addition, our scheme realizes user revocation, ensures the privacy of cloud data, and satisfies unforgeability of tags. Performance analysis results show that it has higher efficiency than similar schemes.**

Keywords: cloud storage; deduplication of ciphertext; data integrity verification; proxy re-signature

I. INTRODUCTION

In recent years, more and more individuals and companies choose to store data in the cloud servers instead of local storage systems [1], which not only alleviates the cost pressure caused by local data storage and maintenance, but also eliminates the need for complex networks and maintenance of software and hardware systems [2]. In the cloud storage system, the user can use and process the data on the cloud server at any time, but the user loses control of the data stored on the remote node [3]. On the one hand, no matter what highly reliable measures the cloud service provider takes, data loss may occur [4-8]. On the other hand, for their own economic interests, cloud service provider (CSP) may deliberately discard some unaccessed or rarely accessed data to save storage space, and claim that the data is still stored correctly in the cloud [9]. Therefore, CSP cannot be fully trusted by users. Ateniese et al. [10] first proposed the PDP mechanism to ensure the integrity of data on untrusted cloud servers. This scheme allows any third party to challenge the cloud server to verify the possession of the data, which greatly expands the application area of the PDP.

Due to the development of big data and cloud computing technology, CSP is also facing the problem of efficiently managing cloud resources and avoiding the waste of cloud resources [11]. In cloud storage, the same data file may be uploaded to the cloud server by different users, resulting in duplicate file copies [12]. Such duplicate data greatly wastes cloud storage space and complicates data management. Therefore, for the cloud storage and processing of big data, it is crucial to solve the problem of duplicate data storage [13]. Xiong et al.[14] reviewed the research progress of data deduplication; Tian et al. [15] proposed a random client-side data deduplication scheme, which realized multi-user ownership management and data sharing by means of a dynamic encryption key tree. , But the scheme does not support user revocation; Fu et al. [16] proposed Client-side secure deduplication scheme for ciphertext data in cloud storage, but did not achieve data integrity verification; Jin et al. [17] proposed a cloud data audit scheme that supports encrypted data deduplication, and does not support multi-user operations; Zhang et al. [18] proposed an integrity audit scheme that supports key update and ciphertext data deduplication, but requires users to participate in key update online, and does not support multiple users operating.

In response to the above problems, we propose a data integrity verification scheme for ciphertext data security and deduplication. In this scheme, users store data in a cloud server in the form of ciphertext, which can solve the problem of data privacy. The scheme uses block tags to verify the ownership of user data to achieve cloud data ciphertext de-duplication. In data integrity verification, a proxy re-signature method is used to ensure that users do not have to participate in the audit process online all the time, which also reduces the amount of calculations on the cloud server. By comparing similar schemes, our scheme has better efficiency.

II. PRELIMINARY

A. The Bilinear Pairing

Let G and G_T be two cyclic groups with prime order p, and g is a generator of G. A bilinear map $e: G \times G \to G_T$ with the follows properties [19]:

- Bilinearity: For any $a, b \in Z_p^*$, there is $e(g^a, g^b) = e(g,g)^{ab} = e(g^b, g^a)$.
- Nondegeneracy: $e(g,g) \neq 1$.
- Computability: e can be calculated effectively.

B. Discrete Logarithm(DL) assumption

Given g and $g^x \in G$, the DL problem on the group G is to calculate x [20].

C. Computational Diffie-Hellman(CDH) assumption

Given t $a,b \in Z_q^*$, a cyclic groups G with prime order p and generator g, and $(g, g^a, g^b) \in G$, the CDH problem on the group is to calculate $g^{ab} \in G$ [21].

III. DATA INTEGRITY VERIFICATION SCHEME OF DEDUPLICATION FOR CLOUD CIPHERTEXTS

A. System Model

The system model of this scheme is shown in Fig. 1. It includes the following four entities.

- User: The Users, including the first upload user and subsequent upload users, upload data to the cloud service and generate data tags and data signatures.
- Third Party Auditor (TPA): The Third Party Auditor verifies the integrity of the data stored in the CSP on behalf of the user.
- Cloud Service Provider (CSP): The cloud servers, with a large amount of storage space and powerful computing resources, provides storage and computing services for Owners.

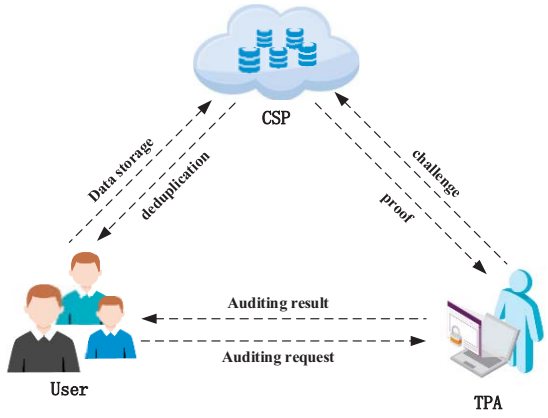

Fig.1 System Model

B. Scheme description

1) Key generation

- System parameters. G and G_T are two cyclic groups with prime order q. p and g denote the generators of G and G_T. Define a bilinear map $e: G \times G \to G_T$ and Two anti-collision hash functions $H_1: \{0,1\}^* \to Z_p^*$ and $h: \{0,1\}^* \to G$. set Public system parameters $params = \{q, G, G_T, p, g\}$.
- Public-private Key generation. The user u_r randomly selects an integer $sk_t \in Z_q$ as the private key and calculates $pk_t = g^{sk_t} \in G$ as the public key, with the corresponding public-private key pair is (sk_t, pk_t).

  TPA randomly selects an integer $x \in Z_q$ as the private key and calculates $P_{pub} = p^x$ as the public key, then the corresponding public-private key pair is (x, P_{pub}).
- Encryption key generation. The first uploading user u_1 divides the file M into n blocks: $M = m_1 \| m_2 \| \cdots \| m_n$,calculates the convergence key $h_i = h(m_i)$ for each data block, and performs privacy protection processing on each data block, which calculate $a_i = h_i \cdot p^{sk_t}$. and then, the user uploads a_i to TPA.

  TPA calculates $b_i = a_i^x$ and sends it the user.

  The user calculates $k_i = \dfrac{b_i}{P_{pub}^{sk_t}}$ and verifies whether k_i is correct by the following equation:

$$e(k_i, p) = e(h(m_i), P_{pub}) \tag{1}$$

  If the verification is passed, the user will use k_i as the encryption key for the data block m_i $(1 \le i \le n)$.
- Re-signature key generation. According to the encryption key k_i and private key sk_t, the user u_r calculates the re-signature key: $rek_i = \dfrac{sk_t}{k_i}$.

2) File initialization

The user u_r uses the encryption key to calculate the ciphertext and file label of the data block, and encrypts the encryption key to ensure the confidentiality of the encryption key. The specific process is as follows:

- The user u_r calculates the block ciphertext $C_i = Enc(k_i, m_i)$ for the data block $m_i (1 \le i \le n)$ of the file M ,then obtains $c_i = H_1(C_i)$.
- The user calculates the label $T_t = H_1(C_1 \| C_2 \| \cdots \| C_n)$ of the file M.
- TPA performs symmetric encryption on the encryption key $k_i (1 \le i \le n)$, that is, calculate $C_{key} = Enc(P_{pub}, k_1 \| k_2 \| \cdots \| k_n)$.
- The user uploads and the label T_t of the file M to the CSP, and the CSP detects whether the file label exists.

3) File storage

When the CSP detects that the file label T_t does not exist, it means that the user who uploaded the file is the first uploading user u_1, the user needs to initialize and upload the file to the CSP.

- Data block initialization. the user needs to calculate the block label and block signature for each data block. The specific process is as follows:

For each file block $m_i(1 \le i \le n)$, calculate the block label $\tau_i = H_1(M) \cdot h(m_i)$ and block signature $\sigma_i = \sigma_i = k_i \cdot (H_i \cdot pk_t^{c_i})^{sk_t}$ Send τ_i and σ_i to the CSP.

The user randomly selects s random numbers $x_1, x_2, \cdots x_s \in Z_q$, calculates $u_j = g^{x_j} \in G(j \in [1,s])$, sets $u = \{u_1, u_2, \ldots, u_s\}$, and sends u to the CSP for storage, where s represents a first-level data block composed of s second-level data blocks.

In order to prevent replay and forgery attacks, we introduced an index table to record the abstract information of the data. The index table is composed of four parts: i , B_i , V_i and IT_i , in which i represents the current block number, B_i represents original block number, V_i represents the current version number of the data block, and IT_i represents the timestamp of the block label. The user u_1 calculates $W_i = \tau_i \| i \| B_i \| V_i \| TI_i$, which is used to represent the summary information of the data block.

The user calculates $T_i = (H_1(W_i) \cdot \prod_{j=1}^{s} u_j^{c_{i,j}})^{k_i}$, which is used for data integrity verification, and sends it to TPA.

- File Upload.

  The user uploads all the data blocks of the file M , all the ciphertexts $C_i(1 \le i \le n)$ and the secret keys C_{key} .

  CSP verifies whether $T_t = H_1(C_1 \| C_2 \| \cdots \| C_n)$ is true. If it is true, it means that the ciphertext and the tag uploaded by the user match.

  CSP verifies whether $c_i = H_1(C_i)$ is established. If it is true, stores T_t , u , τ_i , σ_i , C_i and C_{key} , and adds ID_{user} to ID_M : $ID_M = ID_M \cup ID_{user}$,where ID_M represents the collection of the identity information of all users who own the file M; otherwise, CSP returns an error message.

4) File deduplication

When detecting the existence of a file tag T_t , CSP performs file deduplication.

- File challenge generation. The TPA randomly selects $c(c \le n)$ and generates block index $I = \{s_1, s_2, \cdots s_c\}$. Then generates challenge information $chal = \{i, v_i\}$, in which $i \in I$, $v_i \in Z_p^*$, and sends $chal$ to certifiers.

- File proof generation. The user calculates the response evidence $P_{CV} = \prod_{i \in I} ((k_i \cdot (H_i \cdot pk_t^{c_i})^{sk_t})^{v_i}$ while The CSP calculates its response evidence $\delta = \prod_{i \in I} \delta_i^{v_i}$, then send response evidences to the TPA.

- File proof verification. TPA verifies whether the following equation holds and sends the verification result to CSP.

$$\delta = P_{CV} \tag{2}$$

  If it is true, it means that the user has the same file as the CSP, and the user only needs to send C_{key} and ID_{user} to the CSP. Then CSP stores C_{key} and adds user's identity information: $ID_M = ID_M \cup ID_{user}$.

5) Integrity verification

- Re-signature. The user u_r calculates the re-signature $T_i^{'} = (T_i)^{rek_i}$ and sends it to TPA.

- Challenge generation. The TPA randomly selects $c(c \le n)$, generates block index $I = \{s_1, s_2, \cdots s_c\}$ and calculates a challenge stamp $R = (pk_t)^x$. Then generates challenge information $chal = \{i, v_i, R\}$ and sends it to CSP, in which $i \in I$, $v_i \in Z_p^*$.

- Proof generation. The CSP calculates the linear combination: $MP_j = \prod_{i \in I} v_i \cdot c_{i,j}$, $MP_j \in Z_p^*$ and the hash value of W_i : $H(W_i) = H(\tau_i \| i \| B_i \| V_i \| TI_i)$. The CSP sends the response evidence $P = (DP, H(W_i))$ to the TPA.

- Proof Verify. The TPA calculates $TP = \prod_{(i,v_i) \in I} (T_i^{'})^{v_i}$ and $H_{chal} = \prod_{(i,v_i) \in I} H(W_i)^{r \cdot v_i}$. Then, the TPA verifies that the following equation holds.

$$DP \cdot e(H_{cal}, pk_t) = e(TP, g^r) \tag{3}$$

  If it is true, it means that the CSP has completely saved the user's files without deleting or tampering.

6) Decrypt

When the file M is needed, the user obtains the file M from the CSP through the file tag T_t and the identity ID_{user} .

- First, the user sends the tag T_t of the file M and identity ID_{user} to the CSP.

- After receiving tag T_t and the user identity ID_{user} , the CSP verifies whether T_t and ID_{user} are correct. If the verification passes, sends C_{key} and $C_i(1 \le i \le n)$ to the user; otherwise, return an error message.

- After the user receives C_{key} and $C_i(1 \le i \le n)$, verify whether $T_t = H_1(C_1 \| C_2 \| \cdots \| C_n)$ is established. If it is true, the user uses sk_t to decrypt each block key $k_1 \| k_2 \| \ldots \| k_n = Dec(P_{pub}, C_{key})$, and then uses the block key to decrypt each data block $m_i = Dec(k_i, C_i)$

$(1 \le i \le n)$. otherwise, it means that the CSP has returned the incorrect ciphertext to the user.

C. User revocation

When a user no longer owns a file, the CSP will delete the identity information ID_{user} from user record table ID_M . at the same time, it needs to determine whether the user record table is empty. When the user record table is not empty, no further processing will be performed; When it is empty, this file needs to be deleted from the CSP.

D. Correctness analysis

the correctness of Equations (1), (2) and (3) can be proved as follow:

$$e(k_i, p) = e(\frac{b_i}{P_{pub}^{sk_t}}, p) = e(\frac{a_i^x}{P_{pub}^{sk_t}}, p)$$

$$= e(\frac{(h_i \cdot p^{sk_t})^x}{(\mathrm{p}^x)^{sk_t}}, p) = e(\mathrm{h}_i{}^x, \mathrm{p})$$

$$= \mathrm{e}(\mathrm{h}_i, \mathrm{p}^x) = \mathrm{e}(\mathrm{h}(\mathrm{m}_i), \mathrm{P}_{pub})$$

The above formula transformation shows that Equation (1) holds. Equation (2) can be proved as follows:

$$\prod_{i \in I}(\delta_i^{v_i}) = \prod_{i \in I}((k_i(H_1(M) \cdot pk_t^{c_i})^x)^{v_i})$$

$$= \prod_{i \in I}((k_i \cdot (pk_t^{c_i} \cdot H_1(M))^x)^{v_i})$$

$$= P_{CV}$$

In addition, for Equation (3), we can prove that it is correct by the following equation transformation.

$$e(TP, g^r) = e(\prod_{i \in I}(T_i^{'})^{v_i}, g^r) = e(\prod_{i \in I}(T_i)^{v_i \cdot rek_i}, g^r)$$

$$= e(\prod_{i \in I}(T_i)^{v_i \cdot \frac{sk_t}{k_i}}, g^r) = e((\prod_{i \in I}(T_i)^{v_i \cdot \frac{1}{k_i}})^r, g^{sk_t})$$

$$= e(\prod_{i \in I}((H_1(W_i) \cdot \prod_{j=1}^{s} u_j^{c_{i,j}})^{v_i \cdot r}, pk_t)$$

$$= e(\prod_{i \in I}(H_1(W_i)^{v_i \cdot r}, pk_t) e(\prod_{i \in I}(\prod_{j=1}^{s} u_j^{c_{i,j}})^{v_i \cdot r}, pk_t)$$

$$= e(\prod_{i \in I}(H_1(W_i)^{v_i \cdot r}, pk_t) e(\prod_{i \in I}(\prod_{j=1}^{s} u_j^{c_{i,j}})^{v_i}, pk_t^{r})$$

$$= e(H_{chal}, pk_t) e(\prod_{i \in I}(\prod_{j=1}^{s} u_j^{c_{i,j}})^{v_i}, R)$$

$$= e(H_{chal}, pk_t) \cdot DP$$

TABLE I. CALCULATION OVERHEAD COMPARISON OF FILE STORAGE AND DEDUPLICATION

		Fu scheme [16]	*Our scheme*
File storage	*Data block initialization.*	*2nTmul+Thash*	*2n(Texp+Tmul)*
	File upload	*n(Tp+Tmul+Texp+Thash)*	*(c+1)Thash*
File deduplication	*File proof generation*	*3cTmul+Thash*	*3c(Tadd+Tmul)+cTexp*
	File proof verification	*cTmul*	*Tp*

TABLE II. CALCULATION OVERHEAD COMPARISON OF INTEGRITY VERIFICATION STAGE

		Jin scheme [17]	*Our scheme*
Integrity verification	*Proof generation*	*2c(Texp+Tmul)+c(Tp+Thash)*	*C(Texp+Tmul+Tp+Thash)*
	Proof Verify	*(c+2)Texp+(c+1)Tmul+2Tp*	*(2c+1)Texp+Tmul+2Tp*

IV. SAFETY ANALYSIS

Theorem 1. The scheme satisfies the unforgeability of tags.

Proof: The security model in document [18] can prove its deduplication scheme is unforgeable under the random oracle, and its security depends on the CDH hypothesis. Based on the security model of Document [18], our scheme can be proved to satisfy the unforgeability of tags.

Theorem 2. This scheme satisfies the privacy of data.

Proof: The user's data is stored in the cloud in the form of ciphertext, and the encryption key is held by the user. Since the encryption key of each data block is jointly generated by the user and the TPA, which contains the convergence key of the data block, the user's private key and the TPA private key, it is difficult for an attacker to crack the encryption of the data Key and is impossible to obtain data information.

V. PERFORMANCE ANALYSIS

We assume that the number of blocks of file *M* is n, and the number of data blocks participating in the challenge is *c*. To simplify the description, we use *Texp*, *Tmul*, *Tadd*, *Thash* and *Tp* to represent the time required for an exponential operation, a multiplication operation, addition operation and a hash operation and a bilinear pairing operation. In the file storage and deduplication stages, we compared the time cost of the scheme with the document [16]. As shown in Table Ⅰ, our scheme has higher efficiency during the File upload and File proof verification.

In the Integrity verification stage, we compared the time cost of this scheme with the document [17], as shown in the table Ⅱ, our scheme has higher efficiency.

VI. CONCLUSION

This paper designs a data integrity verification scheme of deduplication for cloud ciphertexts, in order to achieve cloud

ciphertext data deduplication and data integrity verification. In the scheme, the user and TPA jointly generate the encryption key of the data block, which realizes the secondary encryption of the convergence key, making the data encryption more secure. The scheme uses block signatures to verify user's ownership of the data, which achieve deduplication of the cloud data. In the integrity verification process, a public audit and proxy re-signature methods is used to verify data integrity. By comparing similar scheme, our scheme has higher efficiency in data deduplication and integrity verification.

ACKNOWLEDGMENTS

This work was supported in part by the 13th Five-Year Foundation for Educational Science in Gansu Province (GS[2018]GHBGZ084).

REFERENCES

[1] X. Yang, Y. Li, J. Wang, et al. "Revocable identity-based proxy re-signature scheme in the standard model," Journal on Communications, vol. 40, pp. 153-162, 2019.

[2] Q. Wang, C. Wang, J. Li, et al. "Enabling public verifiability and data dynamics for storage security in cloud computing," European Symposium on Research in Computer Security, pp. 355-370, 2019.

[3] J. Li, M. Krohn, D. MazieresD, et al. "SUNDR: Secure untrusted data repository," Proceedings of USENIX Symposium on Operating Systems Design and Implementation, pp. 1-9, 2004.

[4] G.R.Goodson, J.J.Wylie, G.R.Ganger, M.K.Reiter, et al. "Efficient byzantine-tolerant erasure-coded storage," Proceedings of IEEE Computer Society, pp. 135-144, 2004.

[5] V.Kher, Y.Kim, "Securing distributed storage: challenges, techniques, and systems," Proceedings of the 2005 ACM workshop on Storage Security and Survivability, pp. 9-25, 2005.

[6] L.N.Bairavasundaram, G.R.Goodson, S.Pasupathy, and J.Schindler, "An analysis of latent sector errors in disk drives," Proceedings of the 2007 ACM SIGMETRICS international conference on Measurement and Modeling of Computer Systems, pp. 289-300, 2007.

[7] B. Schroeder and G. A. Gibson, "Disk failures in the real world: What does an mttf of 1, 000, 000 hours mean to you." Proceedings of FAST USENIX, pp. 1-16, 2007.

[8] X. Lu , Z. Pan, H. Xian. "An integrity verification scheme of cloud storage for internet-of-things mobile terminal devices," Computers & Security, vol.92, pp.101-116, 2019.

[9] G. Atenises, R. Burns, R. Curtmola, et al. "Provable data possession at untrusted stores," Proceedings of ACM Conference on Computer and Communications Security, pp. 598-609, 2007.

[10] Y. Fan, X. Lin, W. Liang, et al. "A secure privacy preserving deduplication scheme for cloud computing," Future Generation Computer Systems, vol. 101, pp. 127-135, 2019.

[11] L. Wang, B. Wang, W. Song, et al. "A key-sharing based secure deduplication scheme in cloud storage," Information Sciences, vol. 504, pp. 48-60, 2019.

[12] X. Zheng, Y. Zhou, Y. Ye, et al. "A cloud data deduplication scheme based on certificateless proxy re-encryption," Journal of Systems Architecture, vol. 102, pp. 106-116, 2020.

[13] J. Xiong, Y. Zhang, F. Li, et al. "Research Progress of Data Security Deduplication in Cloud Environment," Journal of Communications, vol.37, pp. 169-180, 2016.

[14] G. Tian, H. Ma, Y. Xie, et al. "Randomized deduplication with ownership management and data sharing in cloud storage," Journal of Information Security and Applications, vol.51, pp. 1-12, 2020.

[15] A. Fu, J. Song, X. Su, et al. "Client-side secure deduplication scheme for ciphertext data in cloud storage," Electronic Journal, vol. 45, pp.2863‑2872, 2017.

[16] Y. Jin, X. Gong, H. He, et al. "CDED: Cloud data audit solution supporting encrypted data deduplication," Small microcomputer system, vol.39, pp.1498‑1503, 2018.

[17] X. Zhang, C. Li, Z. Liu. "Anti-key leakage integrity auditing scheme supporting encrypted data deduplication," Journal of Communications, vol.40, pp.95-106, 2019.

[18] B. Kang, L. Si, H. Jiang, et al. "ID-based public auditing protocol for cloud data integrity checking with privacy-preserving and effective aggregation verification," Security and Communication Networks, vol.3, pp.1-9, 2018.

[19] S. Peng, F. Zhou, J. Li, et al. "Efficient, dynamic and identity-based Remote Data Integrity Checking for multiple replicas". Journal of network and computer applications, vol.134, pp. 72-88, 2019.

[20] K. Fan, M. Liu, G. Dong, et al. "Enhancing cloud storage security against a new replay attack with an efficient public auditing scheme," The Journal of Supercomputing, vol.76, pp. 4857-4883, 2020.