Verification Control Algorithm of Data Integrity Verification in Remote Data sharing

Guangwei Xu[1], Shan Li[1], Miaolin Lai[1], Yanglan Gan[1*], Xiangyang Feng[1], Qiubo Huang[1], Li Li[2] and Wei Li[1]
[1]School of Computer Science and Technology, Donghua University
Shanghai, 201620, China
[e-mail: gwxu@dhu.edu.cn, ylgan@dhu.edu.cn]
[2]College of Architecture and Urban Planning, Tongji University
Shanghai, 200092, China
*Corresponding author: Yanglan Gan

Received May 8, 2020; revised June 5, 2021; accepted January 4, 2022; published February 28, 2022

Abstract

Cloud storage's elastic expansibility not only provides flexible services for data owners to store their data remotely, but also reduces storage operation and management costs of their data sharing. The data outsourced remotely in the storage space of cloud service provider also brings data security concerns about data integrity. Data integrity verification has become an important technology for detecting the integrity of remote shared data. However, users without data access rights to verify the data integrity will cause unnecessary overhead to data owner and cloud service provider. Especially malicious users who constantly launch data integrity verification will greatly waste service resources. Since data owner is a consumer purchasing cloud services, he needs to bear both the cost of data storage and that of data verification. This paper proposes a verification control algorithm in data integrity verification for remotely outsourced data. It designs an attribute-based encryption verification control algorithm for multiple verifiers. Moreover, data owner and cloud service provider construct a common access structure together and generate a verification sentinel to verify the authority of verifiers according to the access structure. Finally, since cloud service provider cannot know the access structure and the sentry generation operation, it can only authenticate verifiers with satisfying access policy to verify the data integrity for the corresponding outsourced data. Theoretical analysis and experimental results show that the proposed algorithm achieves fine-grained access control to multiple verifiers for the data integrity verification.

Keywords: Cloud Storage, Data Sharing, Data Integrity Verification, Multiple Verifiers, Verification Control

The work was sponsored by the Natural Science Foundation of Shanghai (Nos. 19ZR1402000 and 17ZR1400200), Shanghai Education and Scientific Research Project (C160076), the National Natural Science Foundation of China (Nos. 61772018 and 61772128), and the Seed Funds of the Key Laboratory of Ecology and Energy-saving Study of Dense Habitat of Ministry of Education (Tongji University)

http://doi.org/10.3837/tiis.2022.02.011 **ISSN : 1976-7277**

1. Introduction

Although cloud computing is attractive as a cost-effective and high-performance model. However, the reliability of cloud infrastructure has aroused data owners' concern because cloud infrastructure often encounters security issues [1-2]. Cloud computing that is deployed by cloud service provider (CSP) is a technical black box for users, and is also convenient for users to use and manage. Unfortunately, the nature of the black box leads to the lack of transparency in the behavior of CSP and regulatory mechanisms for CSP, causing users to distrust cloud service provider [3]. To solve these problems, many solutions have been proposed to verify the integrity of remotely stored data [4-17]. In a data sharing environment, when users download data resources over the Internet, they are very concerned about whether the data resources have been tampered with or damaged. Therefore, data integrity needs to be verified to ensure the availability of the data before data downloading [6]. However, the cloud storage service provided by CSP is not free, and the data owner as a consumer needs to pay money to CSP for resource consumption. Therefore, CSP's resource consumption caused by the data integrity verification will be borne by the data owner. The resource consumption includes transmission and computation overhead of CSP performing remote data integrity verification in the process of users downloading data. From the previous analysis, it can be seen that if CSP allows users without data access rights to verify the data integrity, some unnecessary expenses will be brought to the data owner. Moreover, CSP allows users without data access rights to launch the data integrity verification, making CSP vulnerable to resource exhaustion attacks [18]. According to the existing verification process, after receiving the verification requests from any verifier, CSP consumes its computation resources to generate the verification proof for the corresponding verified data. In this case, it is unreasonable for the data owner to pay for all the verification. This attack has been introduced as economic denial of sustainability (EDOS) [19-21], which will mean that the finance of the data owner is under attack. Although the user cannot recover the entire block from the verification proof generated by CSP during the verification process, the proof generation consumes CSP's computation resources and causes additional overhead for the data owner.

At present, many attribute-based access control algorithms have been proposed. However, in the process of data integrity verification, only the data owner sets the access policy, and then CSP performs access control on the verifiers according to data owner's access policy, leading to some new problems: (a) The data owner and the verifier are likely to conspire to falsify the verification result, making the data integrity verification result unreliable; (b) When a verifier needs to apply for an attribute key, he cannot get the attribute key in time and is necessary to wait for the data owner to distribute the key since the data owner is not always online; (c) Data owner's resources and computation power is limited. When multiple users apply for the attribute key from the data owner, it may cause a serious burden to the data owner. Thus, we propose a verification control algorithm of multiple verifiers (MV-VCP) to resolve these problems in this paper. The main work of this article is as follows:

1) In order to reduce the waste of verification overhead caused by multiple verifiers who do not have access permission to launch the data integrity verification, this paper designs an attribute-based encryption verification control algorithm for multiple verifiers.

2) In the algorithm, data owner and CSP construct an access structure together, and generate a verification sentinel that checks the authority of verifiers according to the access structure.

3) Since the access structure is co-generated by data owner and CSP, CSP cannot know the attributes of the access structure constructed by the data owner, and the data owner outsources the sentry generation operation to CSP.

4) CSP authenticates verifiers, and only the verifiers who meet the access policy can launch the data integrity verification to the corresponding data stored on CSP.

2. Related Work

2.1 Data integrity verification

With the continuous popularity of cloud storage, remote data integrity verification has received more and more attention, since Atenese et al. [7] and Shacham et al. [8] proposed proofable data possession (PDP) and proofs of retrievability (POR) respectively. The main achievements of this field are as follows.

(1) Public verification scheme. Shacham and Waters [8] proposed the compact proof of data availability generated by a publicly verifiable homomorphic scheme based on the BLS signature [9]. Zhang et al. [22] developed a public verification scheme for cloud storage and proposed to use indistinguishable obfuscation algorithms to process data.

(2) Identity-based integrity verification. Yu et al. [5] proposed identity-based integrity verification, using key-homomorphic cryptographic primitives to reduce the complexity of the system and the establishment and management cost of PKI-based public key authentication framework RDIC meter. Yang et al. [10] proposed a public audit protocol for sharing cloud data, supporting identity privacy and identity traceability. Tao et al. [14] designed a data integrity verification scheme to prevent collusion between CSP and revoked group users when sharing data. Li et al. [3] proposed fuzzy identity-based data integrity verification for cloud storage systems, and introduced complex identity-based audits to solve complex key management.

2.2 Access control

The access control algorithm is mainly used to conform to the relevant conventions and the scope of authorized users' access to information resources, and also to ensure that resources are not accessed by illegal users. Due to the large number of users in cloud computing and the complex relationship between roles and permissions, it is more suitable to use attribute-based access control to implement fine-grained access control [23-26]. Attribute-based access control can provide anonymous authentication, and further defines access control strategies based on different attributes of the requester, environment, and data object. Attribute-based encryption schemes [27-29] can meet these requirements. Next, the development of attribute-based encryption and attribute-based access control are explained separately.

(1) Attribute-based encryption scheme

In order to support data owners to perform fine-grained data access control in semi-transparent public cloud storage, attribute-based encryption (ABE) is introduced [28-29]. At present, many encryption schemes based on attribute encryption have been proposed, mainly divided into key policy attribute-based encryption (KP-ABE) [28] and ciphertext policy attribute-based encryption (CP-ABE) [23,29]. Among them, CP-ABE is very practical in public cloud storage [23,29], only users who match the access strategy can decrypt the related ciphertext, which increases the flexibility of the data access control mechanism. Li et al. [24] proposed an attribute-based ICN naming access control scheme, which implements

flexible attribute authorization by setting attribute rankings to achieve comparison between attributes.

(2) Attribute-based access control

Joseph et al. [26] proposed an attribute-based method to identify malicious clients. They deal with basic applications in black boxes and have not completely eliminated attacks at the algorithm and protocol level. To solve this problem, Xue et al. [18] proposed a combination of cloud-side access control and existing data-owner-based CP-ABE access control to ensure that only users who comply with data owner's access strategy can download the corresponding data on CSP. CSP is only responsible for judging whether a user complies with the access strategy, ensuring user's privacy, and preventing user from launching an EDOS attack on the data owner [20-21].

In the existing access control scheme, the data owner encrypts the data using CP-ABE algorithm, and also achieves fine-grained access control. But even if the data owner encrypts the data uploaded to CSP according to the existing scheme, users who do not meet the conditions can still download the data and can launch the data integrity verification. Unauthorized downloads can also reduce security by facilitating offline analysis and leaking information such as data length or update frequency. At the same time, the computation overhead caused by the proof generation in the data integrity verification is relatively large. If the verifier is not constrained before the verification, the malicious verifiers will continue to launch the verification on the CSP. This will increase the data owner's consumption significantly. Therefore, it is extremely important to control the verification.

3. System Model and Problem Statements

3.1 Data integrity verification model

In the data storage service model, it generally contains the data owner (DO), cloud service provider (CSP) and user (User) which can also be called the third party verifier (TPA) as shown in **Fig. 1**. When the user needs to use the data of the data owner, he downloads the corresponding data from CSP. At this time, if the data is corrupted, it will cause great trouble to the user. Therefore, users are very concerned about the integrity of data on CSP. The traditional verification model uses the third-party verification scheme [7] to ensure the fairness of the data verification. However, in reality, there is no real third party verifier to help the data owner verify the integrity of data. Thus, the user who needs to download the data on CSP will become the verifier. Moreover, the resources on CSP are not free to use and CSP needs to perform corresponding calculations while users verify the data integrity. The data owner is also a consumer who purchases CSP's services. Therefore, the cost of the data integrity verification will be borne by the data owner accordingly. Then the data owner will be very concerned about the number of verification costs caused by CSP calculating the data proof. Data integrity verification mainly consists of the following five steps:

(1) $KeyGen(\lambda) \rightarrow (sk, pk)$. The algorithm is executed by the data owner. The data owner enters a security parameter λ, and then outputs a private key sk and a public key pk.

(2) $TagGen(F, sk) \rightarrow \Phi$.The tag generation algorithm is executed by the data owner before uploading the data. Take the encrypted data F and the private key sk as input. The data owner firstly divides the data into n blocks, and then calculates the tags σ_i, $i \in [1, n]$ for each block. The data owner merges these tags into the tag set $\Phi = \{\sigma_i\}_{i\in[1,n]}$. The data owner sends the tag set Φ and data F to the CSP.

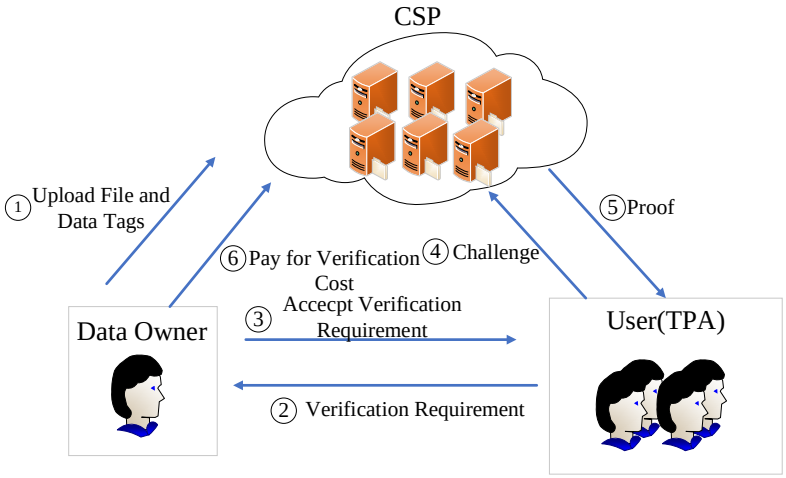


Fig. 1. Data integrity verification model

(3) $Chall(F) \rightarrow C$. The data owner randomly selects the index numbers of some data blocks to be verified, and sends a verification request to TPA. Based on this, TPA generates a corresponding random number for each data block in the information of the extracted data blocks B, and then forms a challenge set $C = \{(i, v_i)_{i \in Q}\}$ to challenge the CSP, where v_i are random numbers.

(4) $ProofGen(F, \Phi, C) \rightarrow P$. CSP responds to the challenge and calculates the verification proof P using the information of the extracted data blocks B stored in his storage space, the data tag Φ corresponding to the extracted data blocks, and the challenge information C provided by TPA, and finally the proof P is returned to TPA.

(5) $VerifyData(C, P, pk) \rightarrow 0/1$. TPA uses the received verification proof P, the public key pk, the challenge information C, and the data tag Φ to determine the integrity of the challenged data blocks and output whether these data blocks are intact or not (i.e., 0 or 1).

3.2 Security Model

The security model of this solution is defined as a selectivity-game between the attacker A and the challenger B. The model is specifically defined as follows:

Init. The adversary A chooses a challenge access structure (M^*, ρ^*), where M^* is an $l^* \times n^*$ matrix, and ρ^* maps each row of M^* to an attribute.

Setup. The challenger runs the Setup algorithm and gives the public parameters PK to the adversary A.

Phase 1. The adversary A issues query for secret keys SK, none of the queried private keys can satisfy the access policy T.

Challenge. The adversary A submits two equal length messages μ_0 and μ_1 to the challenger B. B randomly chooses $b \in \{0,1\}$ and encrypts μ_1 under the challenge access structure (M^*, ρ^*). Finally, it sends the generated challenge ciphertext CT^* to the adversary.

Phase 2. Phase 2 is the same as Phase 1.

Guess. The adversary outputs the guess b of b'. The advantage of A in this game is defined as $Adv_A = \left| Pr(b' = b) - \frac{1}{2} \right|$.

3.3 Problem Statement

(1) Users who do not have data verification rights challenge the data integrity on CSP, causing a waste of verification overhead;

(2) The data owner and the verifier are likely to conspire to falsify the verification results, making the verification results unreliable;

(3) The data owner is not always online. When the verifier needs to apply for the attribute key, it cannot but wait for the data owner to go online for key distribution;

(4) The data owner has only limited resources and computation power. When multiple users apply for the attribute key from the data owner, it may cause a serious burden to him.

4. Verification control algorithm of multiple verifiers

As can be seen from the foregoing, the proof generation requires to consume CSP's resources and incurs corresponding costs for the data owner. Usually, only users with the right to access the data can verify the integrity of data. Users who do not have the right to access data from verifying the data integrity on CSP cause unnecessary overhead to the data owner, and even malicious users continuously launch the data verification on CSP so as to cause EDOS attacks on the data owner. Thus, the user should perform authentication control before the verification. A detailed description of the verification control algorithm for multiple verifiers is as follows.

4.1 Construction of access structure

The access structure is a specific manifestation of the access strategy. Data owner or CSP needs to use a specific access structure to formulate the access strategy. In the verification control algorithm, the access structure is used to authenticate the user. A user can be authorized to verify the data integrity on CSP if and only if his attribute set meets the access structure constructed by CSP and data owner.

Referring to [30] for a (t, n) threshold gate access structure $(P_1, P_2, \ldots, P_n)$, we construct the LSSS matrix M^* on Z_p as

$$M^* = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 2^2 & \cdots & 2^{t-1} \\ 1 & 3 & 3^2 & \cdots & 3^{t-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & n & n^2 & \cdots & n^{t-1} \end{pmatrix}. \tag{1}$$

In this paper, the access structure will be generated by data owner and CSP. The data owner formulates the corresponding access structure $A = (A_1, A_2, 2)$ for the data F, where A_1 is the access structure constructed by data owner, and A_2 is the access structure constructed by CSP. The data owner constructs the LSSS access structure (M, ρ) of access structure A according to formula (1), where matrix M is

$$M = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}. \tag{2}$$

The data owner sets the corresponding access structure A_1 according to the characteristics of data F, and the attribute set $S_1 = \{x_1, \ldots, x_{l_1}\}$. The attribute set S_1 is a set of attributes included in the access policy A_1. The data owner generates the LSSS access structure (M_1, ρ) corresponding to A_1 according to formulas (1) and (2), where the matrix M_1 is

$$M_1 = \begin{pmatrix} a_{0,1} & \cdots & a_{1,n_1} \\ \vdots & \ddots & \vdots \\ a_{l_1,1} & \cdots & a_{l_1,n_1} \end{pmatrix}. \tag{3}$$

CSP formulates corresponding access structure A_2, and attribute set S_2 is a set of attributes contained in the access structure A_2. CSP sets the attribute set $S_2 = \{y_1, \ldots, y_{l_2}\}$, and the corresponding access structure A_2 which is represented by a character string. CSP sends A_2 and S_2 to data owner. CSP generates LSSS access structure (M_2, ρ) according to A_2, where M_2 is

$$M_2 = \begin{pmatrix} b_{1,1} & \cdots & b_{1,n_2} \\ \vdots & \ddots & \vdots \\ b_{l_2,1} & \cdots & b_{l_2,n_2} \end{pmatrix}. \tag{4}$$

After data owner and CSP complete the generation of M_1 and M_2 respectively, the data owner inserts the access structures (M_1, ρ) and (M_2, ρ) into (M, ρ) to form the access structure A. Let l_1 be the number of rows in the matrix M_1, n_1 be the number of columns in the matrix M_1, l_2 be the number of rows in the matrix M_2, and n_2 be the number of columns in the matrix M_2. According to formula (2), the matrix M is calculated as

$$\begin{aligned} M &= \begin{pmatrix} \vec{v}_1 \otimes \vec{u}_1 & \widetilde{M}_1 & 0 \\ \vec{v}_2 \otimes \vec{u}_1 & 0 & \widetilde{M}_2 \end{pmatrix} \\ &= \begin{pmatrix} a_{1,1} & a_{1,1} & a_{1,2} & \cdots & a_{1,n_1} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{l_1,1} & a_{l_1,1} & a_{l_1,2} & \cdots & a_{l_1,n_1} & 0 & \cdots & 0 \\ b_{1,1} & 2 \cdot b_{1,1} & 0 & \cdots & 0 & b_{1,2} & \cdots & b_{1,n_2} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ b_{l_2,1} & 2 \cdot b_{l_2,1} & 0 & \cdots & 0 & b_{l_2,2} & \cdots & b_{l_2,n_2} \end{pmatrix}. \end{aligned} \tag{5}$$

In formula (5), we can see that the data owner inserts the access structure (M_1, ρ) and (M_2, ρ) into (M, ρ), and the computational complexity of the generator matrix M is

$$Cp = n_1 l_1 + n_2 l_2 + l_1 + l_2. \tag{6}$$

The pseudo code of the construction of access structure is shown in Algorithm 1.

Algorithm 1. Access structure generation algorithm

Input: M_1 and M_2**;**
Output: M**;** // The matrix M will be generated according to formula (5)
1. n_1 = column length of M_1, l_1 = row length of M_1;
2. n_2 = column length of M_2, l_2 = row length of M_2;
// DO inserts the LSSS matrix M_1 into matrix M.
3. for $i = 0$ *to* $l_1 - 1$ do
4. for $j = 0$ *to* $n_1 + n_2$ do
5. if $(j = 0)$ then $M[i,j] = M_1[i,0]$; continue;
6. else if $(j > 0$ and $j \le n_1)$ $M[i,j] = M_1[i,j]$;
7. else $M[i,j] = 0$; end if
8. end for
9. end for // CSP inserts the LSSS matrix M_2 into matrix M.
10. for $i := l_1$ *to* $l_1 + l_2 - 1$ do
11. for $j := 0$ *to* $n_1 + n_2$ do
12. if $(j = 0)$ then $M[i,j] = M_2[i,0]$; continue;
13. else if $(j > 1$ and $j \le n_1 + 1)$ $M[i,j] = 0$;
14. else if $(j = 1)$ $M[i,j] = 2 \times M_2[i,0]$;
15. else $M[i,j] = M_2[i - l_1, j - n_1 - 1]$; end if
16. end for
17. end for
18. return M;

For example, assume that the access structure $A_1 = \big((x_1, x_2, x_3, 2), x_4, x_5, 3\big)$ is formulated by the data owner, according to formulas (1) and (2), the LSSS matrix M_1 corresponding to the access structure A_1 is

$$M_1 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 2 \\ 1 & 1 & 1 & 3 \\ 1 & 2 & 4 & 0 \\ 1 & 3 & 9 & 0 \end{pmatrix}.$$

Assuming that the access structure $A_2 = (y_1, y_2, y_3, 2)$ is formulated by CSP, according to formula (1), the LSSS matrix M_2 corresponding to the access structure A_2 is

$$M_2 = \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{pmatrix}.$$

The access strategies A_1 and A_2 constitute a common access structure $A = \Big(\big((x_1, x_2, x_3, 2), x_4, x_5, 3\big), (y_1, y_2, y_3, 2), 2\Big)$. Therefore, according to formula (5), insert M_1 and M_2 into M to form the LSSS access structure (M, ρ) corresponding to the access structure A, the generation process is shown in **Fig. 2**.

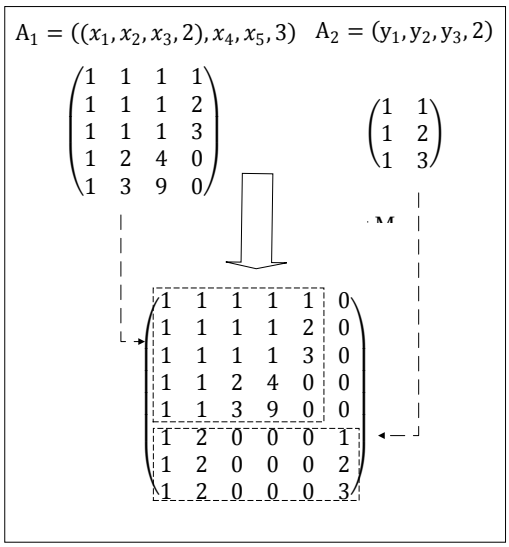


Fig. 2. The generation of matrix *M*

In **Fig. 2**, A_1 is the access structure formulated by data owner, and A_2 is the access structure formulated by CSP. The LSSS matrixes M_1 and M_2 corresponding to structure A_1 and A_2 are generated respectively. Insert M_1 and M_2 into the matrix M to form the LSSS matrix *M* corresponding to the access structure *A*. Finally, the LSSS access structure (M, ρ) is used for subsequent verification control of the user. Only if the attribute set owned by the user satisfies *A*, the user can launch the data integrity verification on CSP.

4.2 Attribute key generation and distribution

In the process of verification control, since data owner is not always online, he needs to outsource his attribute key to CSP, and then CSP replaces the data owner to distribute the key. However, the key distributed by CSP instead of data owner will bring new problems. If data owner directly stores the attribute key on CSP, the CSP can use the corresponding attribute key to unlock the access strategy. Moreover, user sends his attribute set to CSP when applying for the attribute key. The distribution of the attribute key is the only part that leaks the identity information to each attribute authority. Therefore, user needs to hide his attributes when applying for the attribute key from the CSP.

It can be seen from Section 4.1 that the access structure A is constructed by data owner and CSP together, and CSP cannot know the attribute value in the access strategy constructed by data owner. Therefore, the attribute key corresponding to the attribute in the attribute set S_1 is generated by data owner before uploading the data. However, sending the attribute key directly to CSP in plain text will bring new problems. CSP can use the corresponding attribute key to unlock the access policy. At the same time, the attribute values in the attribute set S_1 are generally related to data content or the information of data owner. Publishing the attribute values directly to CSP may cause privacy leakage. Data owner generates an index of the attribute key during the generation of the attribute key, and then encrypts the attribute key before uploading it.

$\boldsymbol{AttKeyGen_{DO}}(S_1)$: It is executed by data owner before uploading the data, with the attribute set S_1 as input. Data owner selects a random number u_1 for $x \in S_1$ to calculate $K_x = h_x^{u_1}$. To prevent CSP from obtaining K_x, data owner will continue to calculate the outsourced attribute key K_x'. For $x \in S_1$, calculate the outsourced key $K_x' = (K_x)^{-r}$, where r

is a random number, and use the attribute value to encrypt the random number r to generate r' by the symmetric encryption algorithm AES, i.e., $r' = AES.Enc(x, r)$. In order to enable user to find the corresponding attribute key on CSP through the attribute value, for all $x \in S_1$, data owner also calculates the index of the attribute key $t_x = H_2(e(H_1(x), g^{sk}))$. Data owner sends $sk_1' = \{K_x', t_x, r'\}_{x \in S_1}$ to CSP.

When a user purchases or registers a service from CSP, he will execute $AttGen_{user}(S')$ to generate an attribute set S', and then send it to CSP. In $AttGen_{user}(S')$, in order to protect user's privacy, user will hide some attributes.

$\boldsymbol{AttGen_{user}}(S')$: It is executed by user and takes the attribute set S' owned by user as input. Generally, it will be run when users register with CSP or purchase services. The user attribute set is $S' = S_1' \cup S_2'$, where S_1' and S_2' are related to S_1 and S_2 in the user attribute set respectively. Referring to the example in Section 4.1, when a user has a purchase behavior on CSP, then one of the user's attribute set is $S = \{x_1, x_2, x_4, x_5, y_1, y_2\}$. Since the privacy of the user is very important, the attributes in the access structure formulated by data owner generally include data information. When the user browses the data or registers in CSP, he does not expect CSP to know what data he has viewed. Therefore, it is necessary to hide the attributes in S_1' in the user attribute set at the stage of user applying for the attribute key. For all $x_i \in S_1'$, calculate $y_i = H_1(x_i)^{uk}$, $pk_u = (g^{sk})^{1/uk}$. Finally, the user sends $Y = \{y_i\}$, pk_u and S_2' to CSP to apply for the attribute key.

After CSP receives $Y = \{y_i\}$, pk_u and S_2', it searches the corresponding attribute key according to the index t_x and the attribute key Y, and then generates the attribute key SK_2 related to S_2' and returns SK to user.

$\boldsymbol{AttKeyGen_{CSP}}(S_2')$: It is executed by CSP and takes the attribute set S_2' as input. CSP searches the corresponding attribute key according to the index of data owner $\{t_x\}_{x \in S_1}$. If

$$t_x = H_2(\mathrm{e}(y_i, pk_u)) \tag{7}$$

holds, CSP saves K_x' and r_x' to SK_1.

CSP selects the random number β and calculates the attribute key SK_2 of the attribute value in S_2' as

$$SK_2 = \left(K = g^{\alpha s^2} g^{ak}, L_1 = g^{ak}, L_2 = h^{ak}, \forall x_i \in S_2' : K_{x_i} = h_{x_i}^{\beta}\right). \tag{8}$$

CSP sends SK_1 and SK_2 to the corresponding users. After CSP returns SK_1 and SK_2, user outsources the attribute key K_x', and the corresponding attribute key K_x can be obtained.

$\boldsymbol{AttKeyGen_{User}}(S_1, SK_2)$: After receiving SK_1 and SK_2, user decrypts r' to obtain $r = AES.Dec(x, r')$, and calculates the attribute key $K_x = (K_x')^r$.

The pseudo code of the attribute key generation and distribution is shown in Algorithm 2.

Algorithm 2. Attribute key generation algorithm

Input: (M, ρ), S_1, S_2, and S';
Output: SK_1, SK_2;

1. $AttKeyGen_{DO}(S_1)$, CSP←SK'_1; // Generating attribute key for data owner
2. $AttGen_{user}(S')$, CSP←{Y, pk_u, S_2'}; // User requests the appropriate attribute key
3. for x *in* S_2'do CSP computes SK_2; end for // Generating attribute key
4. for y_i *in* Y do
5. for t_x *in* SK'_1 do
6. if formula (7) holds then $SK_1 \leftarrow K_x', r_x'$; end if
7. end for
8. end for
9. return SK_1, SK_2;

4.3 Verification authority detection

This paper combines data owner and CSP to jointly generate sentinels for verification control. The access structure (M, ρ) is constructed by data owner and CSP. Neither data owner nor CSP can know the attribute value of the part of the attribute constructed by the other party in the access structure. Thus, the verification sentry needs to be generated in two parts. Specifically, data owner generates the verification sentry ST_{DO} of the access structure constructed by himself and the outsourcing key $p_{s,i}$. CSP generates D_i according to $p_{s,i}$, and then merges ST_{DO} and D_i into the verification sentry ST.

$\boldsymbol{SentinelGen_{DO}}(M, \rho)$: Data owner enters the access structure (M, ρ) to generate a part of the verification sentry ST_{DO}. He randomly selects the secret $s \in Z_p$ and generates a vector $\vec{v} = (s, z_2, \ldots, z_n) \in Z_p$, where $z_2, \ldots, z_n$ are used to share the secret s. For $i \in [1, l_1]$, the data owner calculates $\lambda_i = M_i \cdot \vec{v}$, where M_i is the ith row of the matrix M. He chooses the random number $u_2 \in Z_p$ and calculates ST_{DO} as

$$ST_{DO} = (D = pf \cdot e(g,g)^{as}, D_1' = g^{au_2}, i \in [1, l_1]: D_i = g^{a\lambda_i} h_{\rho(i)}^{u_2}), \tag{9}$$

where $pf \in Z_p$, the function ρ is an injective function, which maps each row in the matrix M to an attribute in the attribute set S, namely $\rho(i) \in S$. ST_{DO} is an intermediate verification sentinel generated by data owner, which is used for user's verification authority. Since s is the key that can be finally controlled by verification, in order to protect data owner's privacy, data owner cannot send the secret s as clear text to CSP. Thus, data owner calculates the outsourcing key $p_{s,i}$ of the secret s by

$$p_{s,i} = g^{a(s \cdot m_{i,1} - m_{i,1})}, \tag{10}$$

where $i \in [l_1 + 1, l_1 + l_2]$ and $m_{i,1}$ is the ith row and first column in the matrix M. Then $hash_{pf} = H(pf)$ is calculated, where $H(\cdot)$ is an anti-collision hash function. Data owner outsources the key $P_s = \{p_{s,i}\}_{\rho(i) \in S_2}$, and the outsourcing vector $\vec{v}' = (1, z_2, \ldots, z_n)$, the verification sentry ST_{DO} and $hash_{pf}$ are uploaded to CSP for storage.

$\boldsymbol{SentinelGen_{CSP}}((M, \rho), \vec{v}', ST_{DO}, P_s)$: CSP inputs access structure (M, ρ), the outsourcing vector $\vec{v}'$, the outsourcing key P_s generated by data owner, and the intermediate verification sentry ST_{DO}. CSP calculates $\lambda_i' = M_i \cdot \vec{v}'$, where $i \in [l_1 + 1, l_1 + l_2]$, M_i corresponds to the ith row in M. CSP chooses random $k \in Z_p$, and calculates D_i by

$$D_i = g^{a\lambda_i'} p_{s,i} h_{\rho(i)}^k = g^{a\lambda_i} h_{\rho(i)}^k, \tag{11}$$

where $\lambda_i = M_i \cdot \vec{v}$. CSP gets sentinel $ST_{CSP} = (D_2' = g^{ak}, i \in [l_1 + 1, l_1 + l_2]: D_i)$.

Finally, the verification sentinel $ST = \langle ST_{DO}, ST_{CSP} \rangle$ is output, and ST will then be used to detect the user verification authority.

$\boldsymbol{AuthProofGen}(SK_1, SK_2, ST)$: The user generates pf' by this algorithm, and proves to CSP that he satisfies the access structure (M, ρ), and can verify the data integrity. Assume that user's attribute set S' meets the access strategy constructed jointly by data owner and CSP. If λ_i is the effectively shared share of secret s, the Lagrange interpolation formula can be used to find a set of coefficients in polynomial time $\{\omega_i \in Z_p\}_{i \in I}$, so that $\sum_{i \in I} \omega_i \lambda_i = s$, where $I = \{i: \rho(i) \in S'\} \subset \{1, \ldots, l\}$. Then user calculates

$$T = \frac{e(D_1' D_2', K)}{\prod_{i \in I} \left(e(D_i, L) e(D_1' D_2', K_{\rho(i)})\right)^{\omega_i}} = e(g,g)^{as}. \tag{12}$$

User calculates $pf' = D/T$ and then sends pf' to CSP to prove that the set of attributes he possesses meets the access structure (M, ρ), so that he can verify the data integrity on CSP.

$\boldsymbol{Verify}(pf', g^{uk})$: After CSP receives pf', it verifies whether user has the authority to verify the data integrity on CSP. CSP verifies whether the attribute set owned by user satisfies the access structure (M, ρ) by

$$hash_{pf} = H(pf^{'}). \tag{13}$$

If formula (13) holds, the attribute owned by user satisfies the access structure, and the verification can be performed; otherwise, CSP rejects user's verification request.

The pseudo code of the process of verification authority detection is shown in Algorithm 3.

Algorithm 3. Verification authority detection algorithm

Input: ST, A;
Output: $true/false$;
1. for $i := 1$ *to* l_1 do
2. ST_{DO}←formula (9); end for // Generating ST_{DO} for data owner.
3. for $i = l_1 + 1$ *to* l_2 do
4. $p_{s,i}$ ←formula (9); end for // Outsourced key $p_{s,i}$
5. CSP←P_s,$\vec{v}^{'}$, ST_{DO}, $hash_{pf}$;
6. for $i = l_1 + 1$ *to* l_2 do
7. D_i←formula (11); end for // Checking sentinel ST
8. CSP generates ST; // Detecting user's authority
9. if user sends S to CSP then CSP sends (ST, A) to user;
10. User computes pf' and sends pf' to CSP;
11. if formula (13) holds then return true;
12. else return false;
13. end if

5. Algorithm Analysis

In this paper, the algorithm constructs the access structure by data owner and CSP to realize the verification control of users, so that users without data access rights cannot launch the data integrity verification for data owner. In order to illustrate the feasibility of the algorithm, the security analysis is conducted in this section, and the theoretical analysis of computational complexity and storage and transmission overhead is implemented in Section 6.1.

The discrete logarithm calculation hypothesis (abbreviated as DL problem) supposes that a $\in Z_p^*$, p is a large prime number, and g_1 is a generator of group G_1, where $g_1^a \in G_1$, $g_1 \in G_1$. Take g_1^a as input and output a.

Definition 1. Discrete logarithm hypothesis (abbreviated as DL hypothesis). It exists $\varepsilon > 0$. The advantage of any attacker in solving the DL problem on group G_1 in a polynomial time algorithm Θ is defined as follows

$$AdvDL_{\Theta} = \Pr\left[\Theta(g_1, g_1^a) = a : a \overset{R}{\leftarrow} Z_p\right] \leq \varepsilon.$$

It can be seen from the above formula that the possibility of solving the DL problem is equivalent to using a random number a to perform a violent collision on Θ, with a probability of $\frac{1}{p} < \varepsilon$. Let p be a sufficiently large prime number. The advantage of solving the DL problem can be ignored, because the solving probability is close to zero. That is to say, it is

computationally difficult or impossible to solve the DL problem on the group G_1 based on the establishment of the above hypothesis [28].

Definition 2. Decision-making q-BDHE hypothesis. Assume that G represents a bilinear group of order p, g and h are two independent generators of the group, select a random value $\alpha \in Z_p$, and then define $y_{g,\alpha,l} = (g_1, g_2, \ldots, g_l, g_{l+2}, \ldots, g_{2l}) \in G^{2l-1}$, where $g_i = g^{(\alpha^i)}$. The algorithm makes a guess based on the output value $z \in \{0,1\}$. If $\left|\Pr\left[\mathrm{B}\left(g, h, y_{g,\alpha,l}, e(g_{l+1}, h)\right) = 0\right] - \Pr\left[\mathrm{B}\left(g, h, y_{g,\alpha,l}, Z\right) = 0\right]\right| \geq \varepsilon$, then the advantage ε is defined to solve the decision-making problem under groups G and G_T. If no polynomial time algorithm solves the decision-making problem with a non-negligible advantage, then the decision-making hypothesis holds in groups G and G_T.

5.1 Robustness of verification control

Users whose attribute set does not satisfy the access structure cannot pass verification control. Suppose that the decision-making q-BDHE hypothesis holds. Without any polynomial time, the adversary can selectively destroy the algorithm by challenging the LSSS matrix.

Init. Suppose adversary A has a non-negligible advantage $\epsilon = Adv_A$ to break this algorithm. Adversary A chooses an access structure (M^*, ρ^*), where M^* has l^* rows and n^* columns.

Setup. The simulator chooses the random number $a' \in Z_p$, and $a = a' + a^{q+1}$, so that $e(g,g)^{\alpha} = e(g^a, g^{a^q})e(g,g)^{a'}$. For $1 \leq x \leq S$, choose a random number z_x. Let X denote the index set i, and $\rho^* = x$. The simulator calculates h_x by $h_x = g^{z_x} \prod_{i \in X} g^{aM^*_{i,1}/b_i} \cdot g^{aM^*_{i,1}/b_i} \ldots g^{aM^*_{i,1}/b_i}$. If $X = \emptyset$, then $h_x = g^{z_x}$.

Phase 1. In this phase, adversary A generates attribute set S. At the same time, adversary A sends S to the simulator to obtain the private key, where S does not satisfy M^*. The simulator selects a random number $r \in Z_p$, and then selects a vector $\omega = (\omega_1, \ldots, \omega_{n^*}) \in Z_p$, where $\omega_1 = -1$. According to the definition of the LSSS matrix, if the attribute set S does not satisfy the access structure (M^*, ρ^*), there must be $\omega \cdot M^*_i = 0$ for any $\rho(i) \in S$. The simulator will define t as $t = r + \omega_1 a^q + \omega_2 a^{q-1} + \ldots + \omega_{n^*} a^{q-n^*+1}$.

And let $L = g^{\mathrm{r}} \prod_{i=1,\ldots,n^*} (g^{a^{q+1-i}})^{\omega i} = g^{\mathrm{t}}$. The simulator calculates K by $K = g^{\alpha'} g^{ar} \prod_{i=2,\ldots,n^*} (g^{a^{q+2-i}})^{\omega i}$.

For $\forall x \in S$, calculate K_x by

$$K_x = g^{(v_x + \beta d_x)t\beta} \cdot \prod_{i \in X} \prod_{j=1,\ldots,n^*} \left(g^{\left(\frac{a^j}{b_i}\right) r\beta^2} \prod_{k=1,\ldots,n^*, k \neq j} \left(g^{a^{q+1+j-k}/b_i}\right)^{\omega_k \beta^2}\right)^{M^*_{i,j}}$$

Challenge. The adversary generates two plaintexts m_0 and m_1, and then sends them to the simulator. The simulator randomly selects $b \in \{0,1\}$, and then calculates $C = m_b T \cdot e(g^s, \alpha')$, $C' = g^s$. Applying the vector $\vec{v} = (s, sa + y_2', sa^2 + y_3', \ldots, sa^{n^*-1} + y_{n^*}') \in \mathbb{Z}_p^{n^*}$, we have

$$C_i = \left(g^{v_{\rho^*(i)}} \cdot H(\rho^*(i))\right)^{\gamma r_i'} \cdot \left(\prod_{j=1,\ldots,n^*} g^{aM_{i,j} y_j}\right) \cdot \left(g^{b_i s}\right)^{-\gamma\left(v_{\rho^*(i)} + d_{\rho^*(i)}\right)} \cdot \left(\prod_{k \in R_i} \prod_{j=1,\ldots,n^*} \left(g^{a^j s (b_i/b_k)}\right)^{\gamma M^*_{k,j}}\right)$$

Phase 2. Repeat phase 1.

Guess. The adversary outputs a guess b' to b. If $b' = b$, the simulator outputs *0* to guess

$T = e(g,g)^{a^{q+1}s}$; otherwise, it outputs 1,and T is one random element of the group G.

The advantage of calculating simulator B to get the correct guess result is $Pr\left[B\left(\vec{y}, T = e(g,g)^{a^{q+1}s}\right) = 0\right] = \frac{1}{2} + Adv_A$.

5.2 Resisting EDOS attacks

The security of many ABE schemes [28-29] and the schemes in this paper are based on the assumption that no probabilistic polynomial time algorithm can solve the q-DBDH problem and has a non-negligible advantage. This assumption is reasonable because the DL problem is widely considered to be tricky in the large number domain [11-12], and the selected group is a cyclic multiplicative group of prime order, where the q-DBDH problem is considered difficult. Therefore, a malicious user cannot challenge a download request to CSP through the malicious user, so that CSP continues to provide downloads, resulting in resource consumption and EDOS attacks. Therefore, in order to prevent this kind of attack, CSP sends a verification sentinel ST to verify whether a user has download permission before data downloading. However, the size of ST is much smaller than the size of data on the CSP. Moreover, the computation overhead of CSP executing the verification authority detection is much smaller than that of CSP generating the integrity proof. Therefore, an EDOS attack cannot be caused.

5.3 Resisting collusion attacks

Because the access structure is constructed together by data owner and CSP. If the malicious users collude with data owner, they only obtain the access structure A_1 and generate the LSSS matrix M_1 referring to formula (3). However, they cannot obtain the access structure A_2 provided by CSP so that they are impossible to generate the LSSS matrix M_2 by formula (4). Therefore, the LSSS matrix M cannot be calculated by formula (5), that is, the malicious users do not have the permission to verify the data integrity on CSP. In a similar manner, if the malicious users collude with CSP, they will not be able to be granted the verification authorization since the LSSS matrix M cannot be calculated by formula (5).

Moreover, even if the malicious users collude with each other, they cannot grant the verification authorization either. The reason is analyzed as follows. When a malicious user obtains SK_1 and SK_2 from CSP, he needs to decrypt the outsourced attribute key $K_x^{'}$ to obtain the final attribute key K_x. However, decrypting the outsourced attribute key, i.e., $K_x^{'}(K_x^{'} = (K_x)^{-r})$, requires first to decrypt $r^{'}$ to get $r(r = \text{AES.Dec}(x, r^{'}))$. But decrypting $r^{'}$ requires the attribute x. Even if malicious users collude with each other to obtain each other's attributes, but it can be known in Section 5.1 that the adversary cannot selectively destroy the algorithm by challenging the LSSS matrix without any polynomial time referring to Definition 2.

6. Performance Analysis

6.1 Theoretical analysis

6.1.1 Computational complexity

Let the multiplication operation consumption in G be Mul_G, exponential operation consumption be Exp_G, and bilinear pairing operation ($e: G \times G \rightarrow G_T$) consumption be $Pair$. The complexity of the algorithm needs to be analyzed from three aspects: the computation overhead generated by data owner, the computation overhead generated by CSP, and the computation overhead generated by user.

(1) Computation overhead of data owner

Data owner executes $SentinelGen_{DO}((M,\rho),PK)$ to generate the verification sentry ST_{DO} and $AttKeyGen_{DO}(S_1)$ to generate the intermediate key before uploading the data F. Assuming that there are n_1 attributes in the attribute set S_1, the calculation of $SentinelGen_{DO}((M,\rho),PK)$ consumes $Exp_G + l_1(2Mul_G + 2Exp_G)$. The computation overhead of $AttKeyGen_{DO}(S_1)$ is $2l_1Exp_G$.

(2) Computation overhead of CSP

CSP calculates C_i and generates the attribute key SK_2 of the attribute set S_2. Suppose there are l_2 attributes in the attribute set S_2. The computation overhead of CSP calculating D_i is $l_2(2Mul_G + 2Exp_G)$. The computation overhead of generating the attribute key SK_2 is $Mul_G + 4Exp_G + l_2Ex_G$.

(3) Computation overhead of user (or TPA)

User's computation overhead is mainly generated by $AuthProofGen(SK,ST)$. At this stage, user generates pf' to prove that the set of attributes he possesses meets the access structure (M,ρ). Suppose there are n attributes in the attribute set owned by the user, and the computation overhead incurred by user at this stage is $nEx_G + (n+1)Pair$.

6.1.2 Storage and transmission overhead

Assume that data owner manages n_{DO} attributes and CSP manages n_{CSP} attributes. Let $|p|$ be the element size of G in Z_p. The storage and transmission overhead of the algorithm is also analyzed from three aspects: storage and transmission overhead generated by data owner, storage and transmission overhead generated by CSP, and the storage and transmission overhead generated by user.

(1) Storage and transmission overhead of data owner

Since data owner needs to store all attribute keys, the storage overhead is n_{DO}. At the same time, he needs to transmit the sentinel and the encrypted attribute key to CSP, which consumes the transmission overhead of $n_{DO} + 1$.

(2) Storage and transmission overhead of CSP

CSP needs to store and generate the attribute key and C_i of the attribute set S_2, and the storage overhead is $n_{DO} + 2 \cdot n_{CSP}$. Assume that n users apply for attribute keys from CSP, and each user applies for n_{TPA} attribute keys, the transmission overhead of CSP is $n \cdot n_{TPA}$.

(3) Storage and transmission overhead of user (or TPA)

Let user have n_{att} attributes. The storage overhead of user is n_{att}. The user needs to send pf' to CSP, and n_{att} attributes need to be sent to CSP to apply for the attribute key. Therefore, the transmission overhead of user is $n_{att} + 1$.

6.2 Simulation

To further analyze the performance of the proposed algorithm, two laptops equipped with Intel core i5-4210M 2.60GHz CPU and a 8GB RAM were used as data owner and user respectively. A service system with 4 core CPU and a 8GB RAM was rented from CentOS Alibaba cloud server to simulate CSP. The experimental code is based on PBC-0.5.14 (pairing-based cryptography library), modified and written with CPABE-0.11. The size of element g in group G is 512 bits, and the length of elements in $Z_p$is 160 bits. The access strategy in the form of $(S_1 \text{ AND } S_2 \text{ AND} \ldots \text{AND } S_n)$ is used to simulate the most complex situation, where S_i is an attribute. Each experiment was repeated 20 times in the same environment and the experimental results were averaged. The proposed algorithm in this paper is called MV-VCP.

In the experiment, the performance of MV-VCP, the algorithms partially outsourced protocol (POP) and fully outsourced protocol (FOP) [18], and CP-ABE [23] were compared.

(1) The computation overhead of data owner at the preprocessing phase

The preparation time refers to the calculation time of data owner before uploading the data to CSP. It is seen from **Fig. 3** that the computation overhead and the number of attributes increase linearly. In **Fig. 3**, CP-ABE almost overlaps with POP and FOP since the running time of CP-ABE is only a few milliseconds longer than that of POP and FOP. The computation overhead of MV-VCP is higher than that of POP, FOP, and CP-ABE. However, in MV-VCP, data owner not only encrypts the data according to the access structure, but also generates an intermediate attribute key before uploading the data. Therefore, the calculation time of MV-VCP at this stage is greater than that of POP and FOP. Moreover, data owner generates the key during the preprocessing stage so that subsequent users will not apply to data owner when applying for the attribute key through the attribute. It is only performed once during the entire verification control process, so it will not cause a serious burden on data owner.

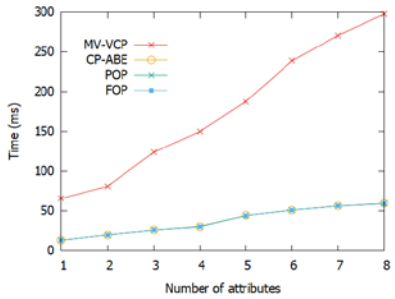


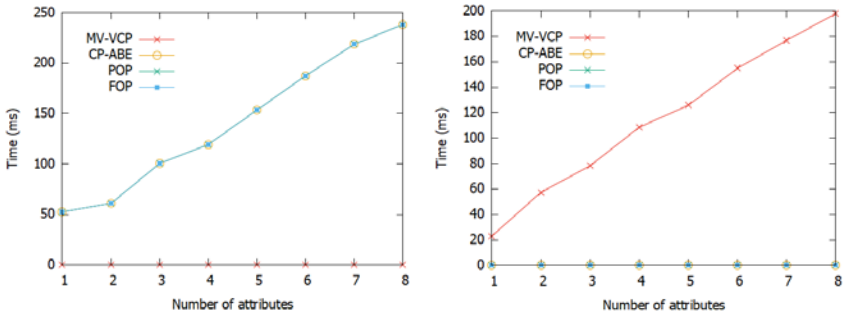
Fig. 3. Data owner preprocessing time

(2) Computation overhead at the key distribution phase

Fig. 4(a) shows the computation overhead incurred by data owner at the key distribution phase. MV-VCP does not require user to apply for the attribute key from data owner, the computation overhead of which is zero. POP requires user to apply for the attribute key from data owner so that the key generation time is linearly related to the number of attributes as shown in **Fig. 4(a)** while only one user applies for the attribute key. FOP consumes a few milliseconds longer than POP and CP-ABE since data owner needs to generate a pair of signature keys for each file in FOP. If multiple users apply for the attribute key, a large amount of computation overhead will be consumed, making data owner vulnerable to resource consumption. Data owner can also generate attribute keys for all attributes in attribute set S_1 in advance. When a user applies for an attribute key from the data owner, the data owner only needs to extract the corresponding attribute key from the previously generated attribute key and send it to the user. In this case, the computation overhead of data owner is also limited.

However, keys distributed by data owner is actually unreasonable. Data owner is not likely to be online always. If a user applies for an attribute key and data owner is not online at that time, the user has to wait until data owner is online. Moreover, if the key is distributed by data owner, when user registers or purchases the service of CSP, CSP needs to forward the request to data owner or user needs to find data owner to apply for the attribute key according to CSP's guidelines. This process is extremely complicated.

The computation overhead of CSP distributing key is shown in **Fig. 4(b)**. Since CP-ABE, POP and FOP will not be distributed by CSP, the computation overhead of POP, FOP and CP-ABE is zero. The overhead of $AttKeyGen_{CSP}(.)$ is still relatively large in **Fig. 4(b)**. When a user applies for an attribute key, CSP calculates the attribute set S_2 attribute key, incurring a large overhead. Therefore, the attribute key *SK* of the attribute set S_2 can be generated by CSP in advance. When a user applies for the key, CSP only needs to search for a key in *SK*.



(a) by DO (b) by CSP

Fig. 4. Key distribution time

(3) Storage and transmission overhead

The storage overhead of attribute keys in MV-VCP is slightly higher than that of POP and FOP, and the overhead of POP is equal to that of FOP as shown in **Fig. 5**. The overhead is linearly related to the number of attributes in access structure *A*. When the number of attributes is 8, even if the size of the attribute key is 2312KB, it does not burden CSP either since the storage resources on CSP are very sufficient.

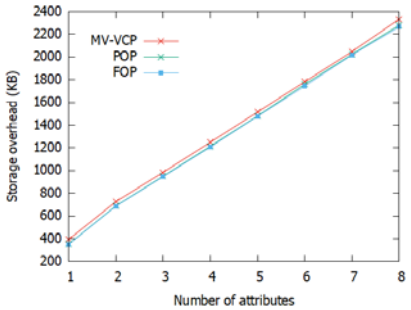


Fig. 5. Storage overhead of attribute keys

The transmission overhead of MV-VCP and POP at the verification authority detection stage is shown in **Fig. 6**. The overhead of MV-VCP is slightly lower than that of POP and FOP. At this stage, CSP sends the verification sentinel *ST* to user to perform authorization detection, where the size of *ST* is linearly related to the number of attributes in the access structure *A*. However, POP sends not only the ciphertext *CT*, but also a challenge of detecting user rights. Moreover, the overhead of FOP is slightly higher than that of POP since FOP has to transmit one more ciphertext of the signature key.

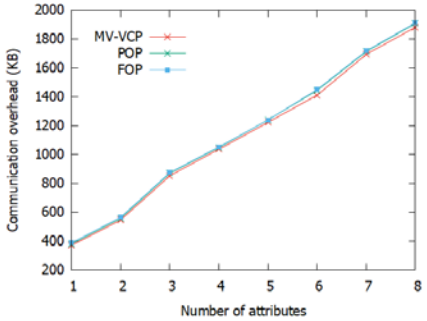


Fig. 6. Transmission overhead of authority

(4) Verification authority detection

Before a user challenges data integrity verification to CSP, his authority needs to be checked first. The experiment set up 10 users to challenge CSP, where each user only launched a challenge. When a user who does not meet the access structure *A* challenges the data on CSP, the computation overhead of CSP performing proof generation is shown in **Fig. 7**. It can be seen that verification control in MV-VCP will greatly reduce the computation overhead of CSP computing unnecessary verification proof relative to the traditional verification algorithm, e.g., DHT-PA [13].

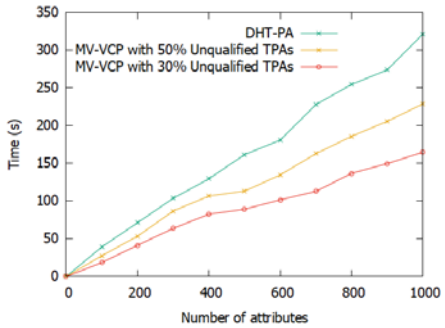


Fig. 7. Proof generation time of unqualified users (or TPAs)

Assume there are 50% of users in the experiment who do not have the authority to verify the data integrity. The experimental results are shown in **Fig. 8**. It can be seen that the effective verification of MV-VCP reaches 100%, while that of DHT-PA is 50%. This is mainly because MV-VCP filters users who have no authority to verify the data integrity on CSP. However, DHT-PA does not perform the authority check on users without the authority.

In summary, MV-VCP can perform verification control on users, construct the access structure by CSP and data owner, and use the access structure to perform verification control on users. The users can challenge the integrity verification of the corresponding data on CSP if and only if the usrs meet the access structure. MV-VCP greatly reduces the computational burden of CSP by removing the unauthorized verification.

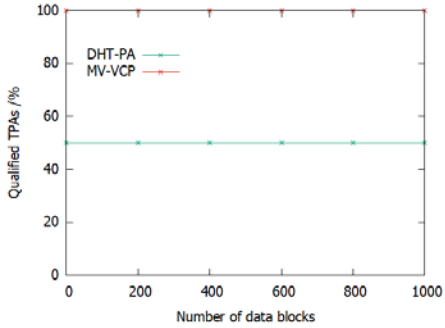


Fig. 8. Effective verification of users (or TPAs)

7. Conclusion

In the process of data integrity verification in cloud storage, users without data access authority perform integrity verification, adding unnecessary verification overhead to data owner. This paper proposes a verification control algorithm. The algorithm mainly includes two aspects. On the one hand, data owner and CSP jointly construct the access structure, which ensures the fairness of the integrity verification results. On the other hand, user hides CSP's attributes during the key distribution stage to ensure user's privacy. The proposed algorithm can effectively intercept users without data access permissions, so that only users who meet the access policy can perform data integrity verification. In the future, we will research the verification control algorithm of multiple data owners.

References

[1] J. Brodkin, "Gartner: Seven cloud-computing security risks," *Infoworld*, vol. 1, no. 1, pp. 1–3, July 2, 2008.

[2] B. R. Kandukuri and A. Rakshit, "Cloud Security Issues," in *Proc. of the 2009 IEEE International Conference on Services Computing*, Bangalore, India: IEEE, pp. 517-520, September 21-25, 2009. Article (CrossRef Link)

[3] Y. Li, Y. Yu, G. Min, W. Susilo, J. Ni and K.-K. R. Choo, "Fuzzy Identity-Based Data Integrity Auditing for Reliable Cloud Storage Systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no.1, pp.72-83, January-February 1, 2019. Article (CrossRef Link)

[4] S. Zawoad, R. Hasan and K. Islam, "SECProv: Trustworthy and Efficient Provenance Management in the Cloud," in *Proc. of the 2018 IEEE INFOCOM*, Honolulu, HI, USA: IEEE, pp. 1241-1249, April 16-19, 2018. Article (CrossRef Link)

[5] Y. Yu, M. H. Au, G. Ateniese, X. Huang, W. Susilo, Y. Dai, G Min, "Identity-Based Remote Data Integrity Checking With Perfect Data Privacy Preserving for Cloud Storage," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 4, pp. 767-778, April, 2017. Article (CrossRef Link)

[6] Y. Deswarte, J.-J. Quisquater and A. Saïdane, "Remote Integrity Checking - How to Trust Files Stored on Untrusted Servers," in *Proc. of the Integrity and Internal Control in Information Systems, Lausanne*, Switzerland: Springer, pp. 1-11, November 13-14, 2003. Article (CrossRef Link)

[7] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, D Song, "Provable data possession at untrusted stores," in *Proc. of the ACM Conference on Computer and Communications Security*, Alexandria, VA, USA: ACM, pp. 598-609, October, 2007. Article (CrossRef Link)

[8] H. Shacham and B. Waters, "Compact Proofs of Retrievability," *Journal of Cryptology*, vol. 26, no. 3, pp. 442-483, July, 2013. Article (CrossRef Link)
[9] D. Boneh, B. Lynn, H. Shacham, "Short Signatures from the Weil Pairing," *Journal of Cryptology*, vol. 17, no. 4, pp. 297-319, September, 2004. Article (CrossRef Link)
[10] G. Yang, J. Yu, W. Shen, Q. Su, Z. Fu and R. Hao, "Enabling Public Auditing for Shared Data in Cloud Storage Supporting Identity Privacy and Traceability," *Journal of Systems and Software*, vol. 113, no. C, pp. 130-139, March, 2016. Article (CrossRef Link)
[11] J. Shen, J. Shen, X. Chen, X. Huang and W. Susilo, "An Efficient Public Auditing Protocol With Novel Dynamic Structure for Cloud Data," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 10, pp. 2402-2415, October, 2017. Article (CrossRef Link)
[12] J. M. Rivas, J. J. Gutiérrez, J. C. Palencia and M. G. Harbour, "Deadline Assignment in EDF Schedulers for Real-Time Distributed Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 10, pp. 2671-2684, October 1, 2015. Article (CrossRef Link)
[13] H. Tian, Y. Chen, C.-C. Chang, H. Jiang, Y. Huang, Y. Chen, J Liu, "Dynamic-Hash-Table Based Public Auditing for Secure Cloud Storage," *IEEE Trans. Services Computing*, vol. 10, no. 5, pp. 701-714, September 1, 2017. Article (CrossRef Link)
[14] T. Jiang, X. Chen and J. Ma, "Public Integrity Auditing for Shared Dynamic Cloud Data with Group User Revocation," *IEEE Transactions on Computers*, vol. 65, no. 8, pp. 2363-2373, August1, 2016. Article (CrossRef Link)
[15] G. Xu, M. Lai, X. Feng, Q. Huang, X. Luo, L. Li and S. Li, "Verification Algorithm for the Duplicate Verification Data with Multiple Verifiers and Multiple Verification Challenges," *KSII Transactions on Internet and Information Systems*, vol. 15, no. 2, pp. 558-579, 2021. Article (CrossRef Link)
[16] G. Xu, S. Han, Y. Bai, X. Feng and Y. Gan, "Data tag replacement algorithm for data integrity verification in cloud storage," *Computers & Security*, vol. 103, no. 3, pp.1-12, 2021. Article (CrossRef Link)
[17] W. Shen, J. Qin, J. Yu, R. Hao and J. Hu, "Enabling Identity-Based Integrity Auditing and Data Sharing with Sensitive Information Hiding for Secure Cloud Storage," *IEEE Trans. Information Forensics and Security*, vol. 14, no. 2, pp. 331-346, February, 2019. Article (CrossRef Link)
[18] K. Xue, W. Chen, W. Li, J. Hong and P. Hong, "Combining Data Owner-Side and Cloud-Side Access Control for Encrypted Cloud Storage," *IEEE Trans. Information Forensics and Security*, vol. 13, no. 8, pp. 2062-2074, August, 2018. Article (CrossRef Link)
[19] J. Idziorek and M. Tannian, "Exploiting Cloud Utility Models for Profit and Ruin," in *Proc. of the IEEE CLOUD*, Washington, DC: IEEE, pp. 33-40, August, 2011. Article (CrossRef Link)
[20] N. Vlajic and A. Slopek, "Web bugs in the cloud: Feasibility study of a new form of EDoS attack," in *Proc. of the GLOBECOM Workshops*, Austin, TX, USA: IEEE, pp. 64-69, December 8-12, 2014. Article (CrossRef Link)
[21] G. Ananthanarayanan, S. Agarwal, S. Kandula, A. Greenberg, I. Stoica, D. Harlan, E Harris, "Scarlett: Coping with skewed content popularity in mapreduce clusters," in *Proc. of the Eurosys*, Salzburg, Austria: ACM, pp.287-300, April, 2011. Article (CrossRef Link)
[22] Y. Zhang, C. Xu, X. Liang, H. Li, Y. Mu and X. Zhang, "Efficient Public Verification of Data Integrity for Cloud Storage Systems from Indistinguishability Obfuscation," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 3, pp. 676-688, March, 2017. Article (CrossRef Link)
[23] B Waters, "Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization," in *Proc. of the Public Key Cryptography*, Taormina, Italy: Springer, pp. 53-70, March 6-9, 2011. Article (CrossRef Link)
[24] B. Li, D. Huang, Z. Wang and Y. Zhu,"Attribute-based Access Control for ICN Naming Scheme," *IEEE Trans. Dependable Sec. Comput*, vol. 15, no. 2, pp. 194-206, March, 2018. Article (CrossRef Link)
[25] T. V. X. Phuong, R. Ning, C. Xin and H. Wu, "Puncturable Attribute-Based Encryption for Secure Data Delivery in Internet of Things," in *Proc. of the INFOCOM 2018*, Honolulu, HI, USA: IEEE, pp. 1511-1519, April 16-19, 2018. Article (CrossRef Link)

[26] J. Idziorek, M. Tannian and D. Jacobson, "Attribution of Fraudulent Resource Consumption in the Cloud," in *Proc. of the IEEE CLOUD*, Honolulu, HI, USA: IEEE, pp. 99-106, June 24-29, 2012. Article (CrossRef Link)

[27] A. Sahai and B. Waters, "Fuzzy Identity-Based Encryption," *EUROCRYPT*, Aarhus, Denmark: Springer, pp. 457-473, May 22-26, 2005. Article (CrossRef Link)

[28] V. Goyal, O. Pandey, A. Sahai and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. of the ACM Conference on Computer and Communications Security*, Alexandria, VA, USA: ACM, pp. 89-98, October, 2006. Article (CrossRef Link)

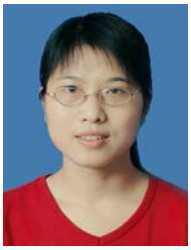
[29] J. Bethencourt, A. Sahai and B. Waters, "Ciphertext-Policy Attribute-Based Encryption," in *Proc. of the IEEE Symposium on Security and Privacy*, Oakland, California, USA: IEEE, pp. 321-334, May 20-23, 2007. Article (CrossRef Link)

[30] Z Liu, Z Cao, "On Efficiently Transferring the Linear Secret-Sharing Scheme Matrix in Ciphertext-Policy Attribute-Based Encryption," *IACR Cryptology ePrint Archive*, January 2010. Article (CrossRef Link).

Guangwei Xu is a professor in the School of Computer Science and Technology at Donghua University, Shanghai, China. His research interests include remote data storage, data integrity verification, privacy protection in data sharing, searchable encryption, QoS and routing of the sensor network and internet of vehicles.

Shan Li is a master candidate in the School of Computer Science and Technology at Donghua University, Shanghai, China. Her main research interests include the verification of data integrity and privacy protection in data sharing.

Miaolin Lai is a master candidate in the School of Computer Science and Technology at Donghua University, Shanghai, China. Her main research interests include the verification of data integrity and searchable encryption.



Yanglan Gan is an associate professor of the School of Computer Science and Technology in Donghua University of China. She received her Ph.D. in computer science from Tongji University of China in 2006. Her research interests are parallel and distributed computing, cloud computing, and big data processing.

Xiangyang Feng is an associate professor in the School of Computer Science and Technology at Donghua University, Shanghai, China. His research interests include the data service, and data security.

Qiubo Huang is an associate professor in the School of Computer Science and Technology at Donghua University, Shanghai, China. His research interests include QoS and routing of the wireless and sensor networks, and data security.

Li Li is an associate professor in the College of Architecture and Urban Planning at Tongji University, Shanghai, China. Her main research interests include the big data processing in architecture and urban planning.

Wei Li is a professor in the School of Computer Science and Technology at Donghua University, Shanghai, China. Her main research interests include the data service and security.