Mobile Networks and Applications (2021) 26:1597–1608
https://doi.org/10.1007/s11036-019-01497-z

An Efficient Scheme of Cloud Data Assured Deletion

Yuechi Tian[1] · Tong Shao[2] · Zhen Li[1,2]

Published online: 6 January 2020

Abstract
With the rapid development of cloud storage technology, cloud data assured deletion has received extensive attention. While ensuring the deletion of cloud data, users have also placed increasing demands on cloud data assured deletion, such as improving the execution efficiency of various stages of a cloud data assured deletion system and performing fine-grained access and deletion operations. In this paper, we propose an efficient scheme of cloud data assured deletion. The scheme replaces complicated bilinear pairing with simple scalar multiplication on elliptic curves to realize ciphertext policy attribute-based encryption of cloud data, while solving the security problem of shared data. In addition, the efficiency of encryption and decryption is improved, and fine-grained access of ciphertext is realized. The scheme designs an attribute key management system that employs a dual-server to solve system flaws caused by single point failure. The scheme is proven to be secure, based on the decisional Diffie-Hellman assumption in the standard model; therefore, it has stronger security. The theoretical analysis and experimental results show that the scheme guarantees security and significantly improves the efficiency of each stage of cloud data assured deletion.

Keywords Cloud storage · CP-ABE · Data privacy · Elliptic curve cryptography (ECC) · Assured deletion

1 Introduction

With the rapid development of cloud storage technology, an increasing number of individuals or enterprises choose to store data in the cloud to reduce data management costs. However, compared with traditional data storage methods, data storage in the cloud is beyond the direct control of the data owner, which renders the security of the data difficult to guarantee. The goal of the data owner when deleting data is to prevent access to the data after the data are deleted [1]. When user data reaches their storage period or the user wants to delete the data stored in the cloud, without an effective data deletion mechanism, unauthorized access, privacy leakage and other security issues may occur and hinder the promotion and development of cloud services [2, 3]. Therefore, achieving the assured deletion of cloud data has become an urgent problem to be solved.

In cloud computing, the ownership of data is separated from the management rights of data. To protect the confidentiality and privacy of data, cloud data must be encrypted before they are outsourced to the cloud [4, 5]. A central advantage of using cryptographic primitives such as symmetric-key encryption is that the safety of a large amount of sensitive data can be reduced to the safety of a very small key [6]. Therefore, the issue of cloud data assured deletion is translated into the secure deletion problem of the corresponding key of the client [7]. A cloud data assured deletion scheme can be divided into two types: assured deletion based on key management and assured deletion based on an access control policy.

Assured deletion schemes based on key management. Perlman [8, 9], first proposed an assured deletion scheme based on files. Xiong et al. proposed a cloud data security self-destruction scheme (ISS) that is based on identity-based encryption (IBE) [10], which divides the data to be protected into different security levels according to different sensitivity levels. The first encryption is performed through the key and then by a coupling and extraction algorithm, the original ciphertext is transformed into a coupled ciphertext, a part of the ciphertext is extracted to convert to the ciphertext, and the remaining part is the encapsulated ciphertext. The encapsulated ciphertext is outsourced to the cloud service provider, and

✉ Tong Shao
15613668123@163.com

Yuechi Tian
tyc@hbu.edu.cn

Zhen Li
93052066@qq.com

[1] School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China

[2] Cyberspace Security and Computer College, Hebei University, Baoding 071002, China

the extracted ciphertext and the IBE-encrypted symmetric key are distributed to the distributed DHT network nodes by the Shamir secret sharing scheme. When the key exceeds the lifetime of the DHT node, the DHT node will automatically delete the key component information to prevent restoration and decryption of the encapsulated ciphertext, and assured deletion of the cloud data is realized. Yao et al. [11] proposed a cloud data assured deletion scheme that is based on bitstream conversion. The scheme divides the file into fixed-size data blocks, each block is converted into a bit stream, the location information of the bit stream and the original data are encrypted and uploaded to the cloud, and the bit stream data and key are saved by the data owner. When data are deleted, the data owner deletes the local stored bitstream data and the key to implement the assured deletion of data.

The assured deletion scheme that is based on key management cannot achieve fine-grained access to ciphertext. When the key of the encrypted data is deleted, all users cannot decrypt the ciphertext again.

Assured deletion schemes based on access control policy. Zhang et al. [12] proposed an assured deletion scheme that is based on ciphertext sampling and partitioning. The original data were encrypted by an encryption algorithm that is based on attributes, and the encrypted ciphertext was sampled. The ciphertext was divided into sampling ciphertext and residual ciphertext. The key and the information of the sampled ciphertext are sent to the trusted third party for management, and assured deletion of the cloud data is realized by deleting the key managed by the trusted third party and sampling the ciphertext information. Although this scheme uses an attribute-based encryption algorithm, it does not implement fine-grained access control. For fine-grained access control problems, Liang et al. [13] proposed an assured deletion scheme that is based on an undo tree, which uses a linear secret sharing scheme and a binary tree as the underlying tool and assigns a set of attributes to each user and assigns a unique identifier. The data owner encrypts the data by an encryption algorithm that is based on attributes. The system administrator disables a user from accessing the data by deleting the corresponding user identifier on the undo tree to achieve assured deletion of the data. Wang et al. [14] proposed an assured deletion scheme that is based on attribute revocation, which is a CP-ABE scheme that supports attribute-level user revocation. In this scheme, once the attribute of a user is revoked, the cloud service provider will randomly select the index that corresponds to the undo attribute to update the ciphertext. After the ciphertext is updated, the user who has revoked the attribute will not be able to decrypt the ciphertext. Xue et al. [15] proposed a cloud data assured deletion scheme that is based on hash tree verification for attribute revocation. The scheme employs a set of attributes to describe the encrypted data and constructs the access policy into the private key of a user. If the user private key with attributes satisfies the access structure of the encrypted data, the user can access the data. When the data in the cloud are deleted, the user can change the attributes in the ciphertext by re-encrypting the ciphertext, which prevents the authorized user from decrypting the ciphertext and implement assured deletion of the cloud data. Zhao et al. [16] proposed a keyless hosted revocable attribute-based encryption scheme in a cloud environment. In this scheme, attribute authority and central control construct a keyless escrow key distribution protocol to solve the key escrow problem using secure two-party computing technology. By updating the attribute version key, an attribute level user can be revoked. The system level user can be revoked via central control. After the attribute is revoked, the user cannot decrypt the ciphertext and realize the assured deletion of the cloud data.

Although the assured deletion scheme that is based on an access control policy solves the problem of fine-grained access control, these schemes use a bilinear pair to implement the attribute-based encryption scheme, which creates the problem of low system efficiency.

In this paper, an efficient scheme of cloud data assured deletion (ESAD) is proposed for schemes that cannot balance efficiency and fine-grained access. This scheme employs simple scalar multiplication in an elliptic curve instead of a complex bilinear pair to implement an attribute-based encryption algorithm [17], which simplifies the calculations during encryption and decryption. The key is generated by the key generator and the attribute authorizer in the attribute key management system. The key generator is responsible for generating the system master key and the public key, and the attribute authorizer is responsible for maintaining a list of user attributes. When the user needs to decrypt, the authorized user sends part of the ciphertext to the attribute authorizer. The attribute authorizer generates the user private key via the attribute list, and partially decrypts the ciphertext, which reduce the decryption overhead of the authorized user. When deleting data, the data owner only needs to generate a random number to replace the attribute value of the attribute that needs to be deleted in the attribute list of the original attribute authorize. The public key of the corresponding attribute value is not updated, which ensure that fine-grained access control can be realized without complicated calculation. Based on the decision Diffie-Hellman assumption, the provable security in the standard model of the scheme is implemented, which shows that the scheme has higher security.

2 Preliminaries

2.1 Elliptic curve cryptography (ECC)

Definition 1(ECC) The elliptic curve in ECC is defined by the following equation:

$$y^2 = x^3 + ax + b(\mathrm{mod}p) \quad \left(4a^3 + 27b^2 \neq 0\right)$$

The safety of ECC is based on the Elliptic Curve Discrete Logarithm Problem (ECDLP). For this problem, the algorithms that have not been effective can obtain the corresponding solutions in a short time as follows.

Let $Ep(a, b)$ be an elliptic curve ($a, b \in GF(p)$) defined on the finite field $GF(p)$. Given the two points P and Q on $Ep(a, b)$, find an integer $s \in GF(p)^*$ to satisfy $Q = sP$.

Compared with RSA, ECC can ensure the same security with a smaller key size as solving ECDLP is more difficult than breaking down integers. The security of the elliptic curve cryptosystem in the meta domain, when the length of p is 160 bits, is equivalent to RSA using a 1024-bit modulus [18]. A short key indicates a small bandwidth and storage requirements, which may be a decisive factor in some applications [19].

2.2 Merkle hash tree (MHT)

The Merkle Hash Tree is an extensively employed authentication structure that can be used to efficiently check if a set of elements are stored unaltered. An MHT is a binary tree where the leaf nodes contain the hashes of authentic data and each internal node stores the hash values of its children nodes. Figure 1 depicts an example of an authentication structure, where the MHT is constructed for an ordered set of data blocks $a_1, a_2, \ldots, a_8$; each internal node and the root node is generated from its two children nodes. For example, $h_2 = h(h_5 \parallel h_6)$, $h_3 = h(h(a_1) \parallel h(a_2))$, and $h_R = h(h_1 \parallel h_2)$. The auxiliary authentication information (AAI) for element a_i represents the sibling nodes of a_i, which are located on the path from the ith leaf node to the root; thus, the root can be computed. To validate the integrity of block a_2, the verifier with the authentic h_R requests the $h_R^{'}$ corresponding auxiliary authentication information $\Omega = (h(a_1), h_4, h_2)$ from the prover, computes $h(a_2)$, $h_3 = h(h(a_1) \parallel h(a_2))$, $h_1 = h(h_3 \parallel h_4)$, and $h_R^{'} = h(h_1 \parallel h_2)$, and checks if the calculated is $h_R^{'}$ equal to h_R.

2.3 Decisional Diffie-Hellman (DDH)

Definition 2 (DHH) The DDH assumptions in the elliptic curve are defined as follows: the challenger selects the cyclic group P of order r, where G is the generator on group P, and randomly selects the parameters $a, b \in Z_r$. If the Challenger gives the adversary parameters (G, aG, bG), the adversary cannot easily distinguish $abG \in P$ from the random element R in group P. *Simulator* B guesses through the output, defines its advantage ε to solve the DDH hypothesis with group P, if

$$\left| \begin{matrix} Pr[\mathrm{B}(G, aG, bG, Z = abG) = 0] - \\ Pr[\mathrm{B}(G, aG, bG, Z = R) = 0] \end{matrix} \right| \geq \varepsilon,$$

If any polynomial time adversary has at most a negligible advantage to solve the DDH problem, then we assume that DDH is true in group P.

3 System model and security model

In this section, we introduce the system model of the ESAD scheme and the security assumptions.

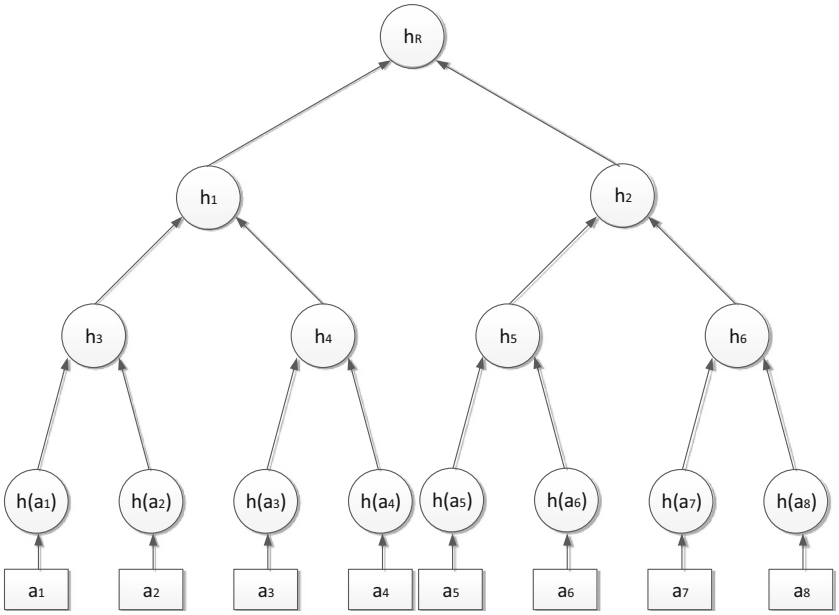


Fig. 1 An example of MHT

3.1 System model

The system model of the ESAD solution is shown in Fig. 2. The model includes four types of entities: Attribute Key Management System (AKMS), Data Owner (DO), Cloud Service Provider (CSP), Authorized user (AU).

AKMS is responsible for the generation and update of the key and helps an authorized user to decrypt part of the ciphertext. The main internal structure is shown in Fig. 3. AKMS is primarily composed of a key generator and an attribute authorizer. The two parts independently communicate with other components and the TPM security chip protects its internal data. The key generator is responsible for managing the system master key and the system public key. The attribute authorizer is responsible for assigning each authorized user a unique identifier ID and maintaining a list of attributes for the authorized user. When the authorized user decrypts the ciphertext, the attribute authorizer generates the user private key and assists the authorized user to partially decrypt the ciphertext. A question-and-answer method is employed between the key generator and the attribute authorizer to confirm that each is normally functioning. The attribute authorizer periodically issues a question to the key generator, and the key generator must return the correct answer within the specified time. If a question or answer is not returned within the specified time, the party is considered to have an abnormality, and the other party will replace the abnormal party to handle the transaction, which ensures normal operation of the system and prevents system paralysis caused by failure of a single point.

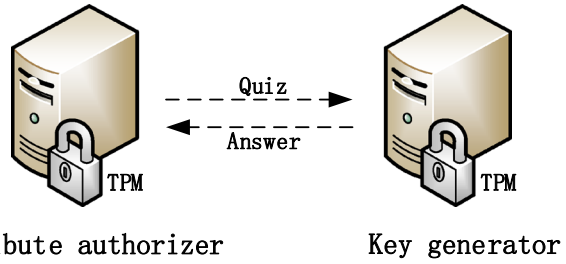


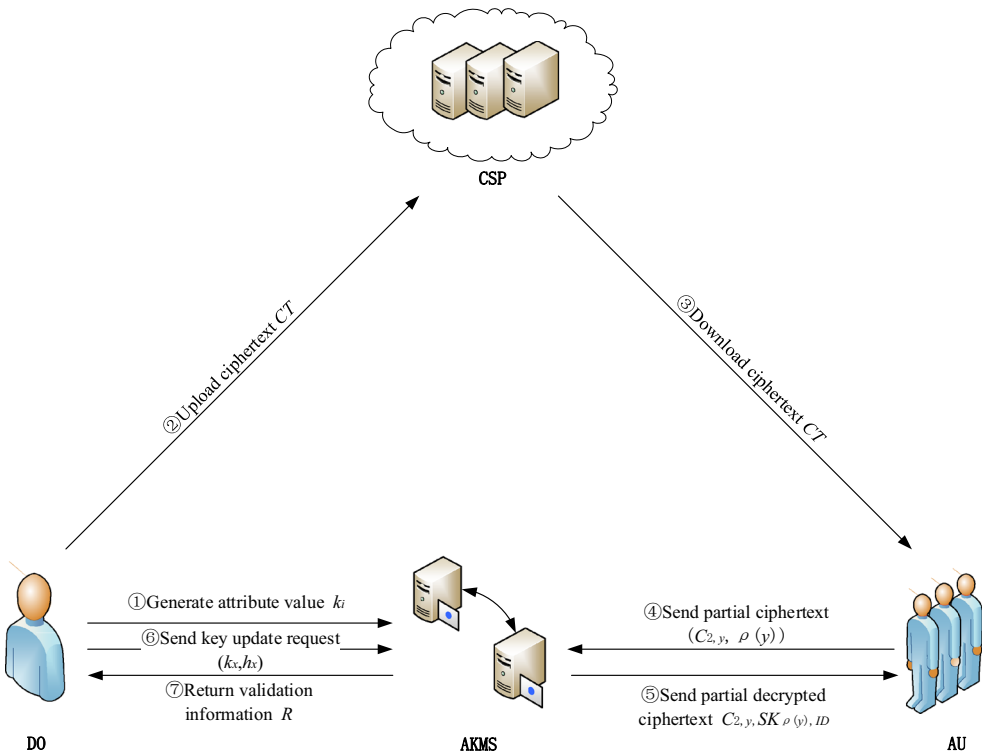
Fig. 3 Internal structure of attribute key management system

The CSP is responsible for storing the encrypted data of the data owner. CSP honestly stores the encrypted data of the data owner and reliably responds to data requests.

The DO is responsible for assigning the unique identifier k_i to each attribute and encrypting the data that needs to be uploaded to the cloud. When the cloud data are deleted, the DO sends an update request to the attribute authorizer in the attribute key management system and verifies the returned update result.

The AU accesses the encrypted data stored by the data owner in the cloud and obtains the encrypted data by sending a data request to the cloud service provider. After obtaining the encrypted data, the AU requests that the attribute authorizer in the attribute key management system partially decrypt the ciphertext. After receiving the result returned by the

Fig. 2 The system model



attribute authorizer, the UA decrypts all ciphertexts to obtain the original data.

3.2 Security assumptions

Based on the system model, the following five assumptions are made:

1). CSP is untrustworthy. While providing data storage services to data owners, it may store data that is stored in the cloud by a data owner for backup or steal data content without authorization from a user and disclose the data to unauthorized individuals or organizations.
2). AKMS is semi-trusted. The attribute authorizer in AKMS will honestly assist an authorized user in decrypting the encrypted data but will not faithfully execute the key update request of the data owner. The attribute authorizer retains only the latest attributes list of the authorized user for the convenience of management.
3). AKMS is unbreakable. The system environment in AKMS is sufficiently secure to withstand external attacks and protect the contents of AKMS from being stolen and tampered by malicious users.
4). The AU is trusted and does not actively disclose the plaintext data of a data owner to an unauthorized entity or actively disclose the obtained decrypted information.
5). A secure channel exists between the entities. The secure channel is established by the shared key obtained by the parties via a key agreement between two parties, and can also be established based on the public-private key encryption and decryption between two parties.

3.3 Security model

This section defines a selective security model of standard semantic security, namely, ciphertext indistinguishability under chosen plaintext attacks (IND-CPA).

1). **Initialization.** Adversary A chooses a challenge access structure $(\boldsymbol{A}, \rho)$ and then sends it to challenger B.
2). **Setup.** Challenger B runs the *Setup* algorithm to generate the necessary public parameters for the system and the public and secret key pair for each attribute. The challenger sends the public key *PK* to adversary A.
3). **Phase 1.** Adversary A can adaptively send the attribute secret key pairs to the challenger, such as (S_1, ID_1), (S_2, ID_2), $\cdots$, (S_n, ID_n), with the restriction that any set of keys can-not satisfy the access control structure $(\boldsymbol{A}, \rho)$.
4). **Challenge.** Adversary A submits two messages M_0 and M_1 of equal length to Challenger B. Challenger B randomly selects $\beta \in \{0, 1\}$ and sends the encryption of M_β under the access control structure $(\boldsymbol{A}, \rho)$ to adversary A.
5). **Phase 2.** The adversary may adaptively submit additional key queries (S_{n+1}, ID_{n+1}), (S_{n+2}, ID_{n+2}), $\cdots$, (S_{n+m}, ID_{n+m})) to the challenger with the same restriction indicated in Phase 1.
6). **Guess.** Adversary A may output the value $\beta' \in \{0, 1\}$ as a guess for β. If $\beta' = \beta$, we consider that adversary A wins the game. The adversary advantage in this game is defined as follows:

$$Ad = \left| Pr\left[\beta' = \beta\right] - \frac{1}{2} \right|.$$

If any polynomial time adversary has at most a negligible advantage to win this security game, then we consider that the encryption scheme is safe.

4 Detailed design of the ESAD scheme

4.1 Design idea of ESAD

The design idea of the ESAD scheme. Establish a secure and efficient cloud data assured deletion system. To improve the performance of the encryption algorithm, simple scalar multiplication in an elliptic curve is employed instead of the complex bilinear pair, which simplifies the calculation in the process of encryption and decryption, improves the encryption and decryption efficiency of the data, and ensures the security of the encrypted data. The LSSS [20, 21] is used to split the encryption key, combine with encrypted data to form ciphertext and upload it to the cloud. To prevent collusion attacks, each user is assigned a global unique identification *ID*, each attribute of the user will be bound with the *ID* of the user as the private key of the user, and the user attribute set is maintained by the attribute authorizer in the form of a list. The attribute list of the attribute authorizer is changed by updating the attribute, and the update verification is performed; thus, the cloud-encrypted data cannot be smoothly decrypted, and the assured deletion of the cloud data is realized.

4.2 Algorithm description of ESAD scheme

The workflow of the ESAD scheme is shown in Fig. 2. The main algorithms are described as follows:

Step 1. **System Initialization**

Let $GF(p)$ be a finite field of order p, E be an elliptic curve defined in the finite field $GF(p)$, r be the order of the subgroup in the elliptic curve E, and G be a base point in the subgroup. G generates a cyclic subgroup of elliptic curves E; Z_r is an

integer domain of order r; and the one-way function $H:\{0,1\}^{*}\rightarrow Z_r^{*}$ is randomly selected: the user ID is mapped to the integer domain Z_r.

$$Setup(n,S,k_i)\rightarrow(MSK,PK)$$

For each attribute i in the system attribute set S, the data owner randomly selects the parameters $k_1, k_2, \cdots, k_S \in Z_r$ to upload to the attribute key management system.

The key generator randomly selects n ($n \in Z_r$) to generate the system master key.

$$MSK = (k_1, k_2, ..., k_S, n, G),$$

Generate the system public key parameters.

$$PK = (k_1 G, k_2 G, \cdots, k_S G).$$

The attribute authorizer maintains a list of attributes that correspond to its ID for each user in the system, as shown in Table 1.

Step 2. **Generate a private key**

$$AKMSKeyGen(ki, ID, n) \rightarrow \{SK_{i,\ ID}\}$$

For each attribute i in the user ID attribute set S_{ID}, the attribute authorizer calculates the user private key

$$SK_{i,ID} = k_i + H(ID)n.$$

Step 3. **Data Encryption**

The data owner encrypts the original data M to generate the encrypted data CT, and the encryption algorithm proceeds as follows:

$$Encrypt(M, s, PK_i, \mathbf{A}, \mathbf{v}, \mathbf{u})\rightarrow CT$$

The data owner randomly selects $s \in Z_r$ as the encryption key; it calculates

$$C = M + sG.$$

The key s is split using the LSSS as follows:

Define the access control strategy $(\boldsymbol{A}, \rho)$, where $\boldsymbol{A}$ is a linear secret sharing matrix, ρ is the mapping function of the matrix row vector and the attribute, and ρ maps each row $\boldsymbol{A}_x$ of the matrix $\boldsymbol{A}$ to the attribute $\rho(x)$. As in [22], ρ does not map two different rows to the same attribute. Randomly select the parameters $v_2, v_3, \cdots, v_l \in Z_p$, define the vector $\widetilde{v} = (\widetilde{s}, \widetilde{v}_2, \widetilde{v}_3, ...\widetilde{v}_l)^{\mathrm{T}}$, and calculate the inner product $\lambda_x = \mathbf{A}_x \cdot \mathbf{v}$ for each row $\boldsymbol{A}_x$ of the matrix $\boldsymbol{A}$. Randomly select the parameters $u_2, u_3, \cdots, u_l \in Z_p$, define the vector $\mathbf{u} = (0, u_2, u_3, \cdots, u_l)^{\mathrm{T}}$, and calculate the inner product $\omega_x = \mathbf{A}_x \cdot \mathbf{u}$ for each row $\boldsymbol{A}_x$ of the matrix $\boldsymbol{A}$.

The data owner outputs the ciphertext

$$CT = \left((\mathbf{A}, \rho), C, \left\{C_{1,x} = \lambda_x G + \omega_x PK_{\rho(x)}, C_{2,x} = \omega_x G\right\}_{x=1}^{m}\right).$$

Step 4. **Data decryption**

$$Decrypt\left(C, SK_{\rho(y),ID}\right)\rightarrow M$$

To decrypt the cloud data, the authorized user sends a ciphertext request to the cloud. After obtaining the ciphertext, the authorized user sends the ID and the $(C_{2,\ y}, \rho(y))$ associated with each attribute $y \in S_{ID}$ to the attribute authorizer. The attribute authorizer verifies the sender identity and the attribute set based on the maintained attribute list. If the request is valid, the attribute authorizer calculates each of the requests $(C_{2,\ y}, \rho(y))$

$$\begin{aligned} C_{2,y} SK_{\rho(y),ID} &= \omega_y G\left(k_{\rho(y)} + H(ID)n\right) \\ &= \omega_y k_{\rho(y)} G + \omega_y H(ID)nG \end{aligned}.$$

The attribute authorizer sends the calculated result to the authorized user, and the authorized user receives the returned result and then calculates

$$\begin{aligned} &C_{1,y} - C_{2,y} SK_{\rho(y),ID} \\ &= \left(\lambda_y G + \omega_y PK_{\rho(y)}\right) - \left(\omega_y k_{\rho(y)} G + \omega_y H(ID)nG\right). \\ &= \lambda_y G - \omega_y H(ID)nG \end{aligned}$$

The authorized user selects the vector $\mathbf{c} \in Z_r$, lets $\mathbf{c} \cdot \mathbf{A}_y = (1, 0, \cdots, 0)$, and calculates

$$\sum_{y \in S_{ID}} c\left(\lambda_y G - \omega_y H(ID)nG\right) = sG.$$

Table 1 User attribute list

ID_1	ID_2	…	ID_n
$k_{Att_{11}}$	$k_{Att_{21}}$	…	$k_{Att_{n1}}$
$k_{Att_{12}}$	$k_{Att_{22}}$	…	$k_{Att_{n2}}$
$k_{Att_{13}}$	$k_{Att_{23}}$	…	$k_{Att_{n3}}$
…	…	…	…

The authorized user calculates

$$C-sG = M.$$

The original data M are obtained.

Step 5. **Update Key**

$$Update(ID_{RL_x}, h_x, k_x, n) \rightarrow \left(PK_x, \{SK_{x,ID_{RL_x}}\}, CT'\right)$$

For attribute x, the data owner randomly selects $h_x \in Z_r$ and $h_x \neq k_x$, and sends k_x and h_x to the key generator and attribute authorizer in the attribute key management system. The key generator updates the public key:

$$PK_x = h_x G.$$

The attribute authorizer searches for the user set RL_x that has the attribute x according to k_x and updates the corresponding attribute value to h_x. The new user private key of the attribute x is

$$SK_{x,ID_{RL_x}} = h_x + H(ID_{RL_x})n.$$

The data owner randomly selects $\widetilde{s} \in Z_p$, uses the same access control strategy $(\boldsymbol{A}, \rho)$ as previously defined, randomly selects the parameters $\widetilde{v}_2, \widetilde{v}_3, \cdots, \widetilde{v}_l \in Z_p$, defines the vector $\widetilde{v} = (\widetilde{s}, \widetilde{v}_2, \widetilde{v}_3, ..., \widetilde{v}_l)^T$, and computes the inner product $\widetilde{\lambda}_x = \mathbf{A}_x \cdot \widetilde{v}$ for each row $\boldsymbol{A}_x$ of matrix $\boldsymbol{A}$. The data owner then randomly selects the parameters $\widetilde{u}_2, \widetilde{u}_3, \cdots, \widetilde{u}_l \in Z_p$, defines the vector $\widetilde{\mathbf{u}} = (0, \widetilde{u}_2, \widetilde{u}_3, \cdots, \widetilde{u}_l)^{\mathrm{T}}$, and calculates the inner product $\widetilde{\omega}_x = \mathbf{A}_x \cdot \widetilde{\mathbf{u}}$ for each row $\boldsymbol{A}_x$ of the matrix $\boldsymbol{A}$. The data owner outputs the updated ciphertext

$$CT' = \left((A, \rho), \widetilde{C} = M + \widetilde{s}G, \left\{\widetilde{C}_{1,x} = \widetilde{\lambda}_x G + \widetilde{\omega}_x PK_{\rho(x)}, \widetilde{C}_{2,x} = \widetilde{\omega}_x G\right\}_{x=1}^{m}\right)$$

Step 6. **Delete Data**

$$DeleteDate(ID_{RL_x}, k_x, h_x, n) \rightarrow \{SK_{x,ID_{RL_x}}\}$$

The data owner sends a key update request to the attribute authorizer. First, randomly select $h_x \in Z_r$ and $h_x \neq k_x$, and then send the original attribute value k_x of the attribute x and the updated attribute value h_x to the attribute authorizer. The attribute authorizer finds the user set RL_x, owns the attribute according to k_x, and then replaces k_x with h_x. The user private key after the attribute value is replaced is

$$SK_{x,ID_{RL_x}} = h_x + H(ID_{RL_x})n.$$

The public key remains

$$PK_x = k_x G.$$

When the authorized user in the user set RL_x decrypts the ciphertext, the attribute authorizer calculates $(C_{2,\,x}, \rho(x))$ for each attribute $x \in S_{ID_{RL_x}}$.

$$\begin{aligned} C_{2,x}SK_{\rho(x),ID_{RL_x}} &= \omega_x G\left(h_{\rho(x)} + H(ID_{RL_x})n\right) \\ &= \omega_x h_{\rho(x)} G + \omega_x H(ID)nG \end{aligned}.$$

Return the result to the authorized user. After the authorized user receives the result, it calculates

$$\begin{aligned} &C_{1,x} - C_{2,x}SK_{\rho(x),ID_{RL_x}} \\ &= \left(\lambda_x G + \omega_x PK_{\rho(x)}\right) - \left(\omega_x h_{\rho(x)} G + \omega_x H(ID_{RL_x})nG\right) \\ &= \left(\lambda_x G + \omega_x k_{\rho(x)} G\right) - \left(\omega_x h_{\rho(x)} G + \omega_x H(ID_{RL_x})nG\right) \\ &\neq \lambda_x G - \omega_x H(ID_{RL_x})nG \end{aligned}.$$

Therefore, the ciphertext cannot be decrypted, and the deletion operation of the encrypted data is realized.

Step 7. **Verification**

$$Verify(R, X_{RL_x}, \Omega_{RL_x}) \rightarrow result$$

The attribute authorizer hashes the attribute list of each user in units of users, and then uses the hash value X_{ID} of each column of the attribute list as the leaf node of the hash tree to establish a hash tree and obtain the hash tree root value R. The data owner also maintains a list of attributes of an authorized user, uses the same hash function as the attribute authorizer to calculate the hash values $\widetilde{X}_{ID}$ of each attribute list of the user, and generates a hash tree. When the data owner verifies whether the attribute authorizer performs an update or delete operation, the root value R of the updated attribute list hash tree is sent to the data owner by requesting the attribute authorizer, and the data owner updates the hash value $\widetilde{X}_{RL_x}$ of the user set RL_x. Calculate the local hash tree root value $\widetilde{R}$ using the hash value $\widetilde{X}_{RL_x}$ and the auxiliary authentication information Ω_{RL_x}. If $\widetilde{R} = R$, the attribute update operation requested by the data owner is successful; otherwise, the request execution fails.

5 Security certification

5.1 Choose plaintext attack

Theorem 1 If the PPT adversary A that can break the proposed scheme in this paper with the non-negligible advantage $\varepsilon = Ad$ exists, then the PPT algorithm B that

can overcome the ESAD scheme with the advantage $\frac{\varepsilon}{2}$ exists.

Let G be the generator of group P of order r. Randomly select a, $b \in Z_r$, $\beta \in \{0, 1\}$ and $R \in P$. If $\beta = 0$, then $Z = abG$; otherwise, when $\beta = 1$, $Z = R$. We will construct simulator B to break the deterministic DDH assumption.

1) **Initialization.** Adversary A chooses a challenge access structure $(\boldsymbol{A}, \rho)$ and sends it to B.
2) **Setup**. Simulator B randomly selects $k_i \in Z_r$ for each attribute i and calculates $PK_i = k_i aG$ as a public key.
3) **Phase 1.** Adversary A adaptively performs a key generation query for the user identity ID_j and the user attribute set S_j to B. According to the rule, for each identity ID_j, if S_j satisfies the access control structure $(\boldsymbol{A}, \rho)$, then $\perp$ is output. Otherwise, B responds by recording this attribute i on the attribute list that corresponds to ID_j. For each $j \in S_i$, B computes $SK_{j,ID_j} = k_j a + H(ID_j)n$ as its secret key, and then sends it to A.
4) **Challenge.** Adversary A submits ciphertexts M_0 and M_1 of the same length to B. B randomly chooses the parameters and $s \in Z_r$, then generates the challenge ciphertext $C = M_\beta + sG$. B randomly selects the parameters v_2, v_3, $\cdots$, $v_l \in Z_p$, defines the vector $\mathbf{v} = (s, v_2, v_3, \cdots, v_l)^{\mathrm{T}}$, and calculates the inner product $\lambda_x = \mathbf{A}_x \cdot \mathbf{v}$ for each row $\boldsymbol{A}_x$ of the matrix $\boldsymbol{A}$. B randomly selects the parameters u_2, u_3, $\cdots$, $u_l \in Z_p$, defines the vector $\mathbf{u} = (0, u_2, u_3, \cdots, u_l)^{\mathrm{T}}$, and calculates the inner product $\omega_x = \mathbf{A}_x \cdot \mathbf{u}$ for each row $\boldsymbol{A}_x$ of the matrix $\boldsymbol{A}$. B generates the challenge ciphertext $CT = (, -$
$(A, \rho), C, \left\{C_{1,x} = \lambda_x G + \omega_x k_{\rho(x)} Z, C_{2,x} = \omega_x bG\right\}_{x=1}^{m})$
and sends to it adversary A.
5) **Phase 2.** As described in Phase 1, adversary A may submit additional key queries (i, ID) without violating the constraints.
6) **Guess.** Adversary A outputs the guess β' of β. If $\beta = \beta'$, B outputs 0 to indicate that $Z = abG$ in the previous game; otherwise, B outputs 1 to guess $Z = R$. If $Z = abG$, then CT is a real ciphertext; thus, this process concludes that

$$Pr[\mathrm{B}(G, aG, bG, Z = abG) = 0] = \frac{1}{2} + \varepsilon.$$

When $Z = R$, M_β is completely random to the attacker, which yields

$$Pr[\mathrm{B}(G, aG, bG, Z = R) = 0] = \frac{1}{2}.$$

The advantage to break this security games is

$$\begin{aligned} Ad &= \frac{1}{2}(\Pr[B(G, aG, bG, Z = abG) = 0] + \Pr[B(G, aG, bG, Z = R) = 0]) - \frac{1}{2} \\ &= \frac{1}{2}\left(\frac{1}{2} + \varepsilon + \frac{1}{2}\right) - \frac{1}{2} \\ &= \frac{\varepsilon}{2} \end{aligned}.$$

Therefore, B can simulate a deterministic DDH hypothesis with a non-negligible advantage.

5.2 Collusion attack

To ensure the security of ciphertext in the cloud, the ESAD scheme defends against collusion attacks by introducing user ID into the private key of the users. To prove that the ESAD scheme can resist collusion attacks, assume that Alice intends to collaborate with Bob to decrypt the ciphertext in the access structure $A \wedge B \wedge (C \vee D)$. Alice only has attributes A and B, and Bob only has attribute C. *Alice* and Bob cannot separately decrypt the ciphertext, but if they collude with each other, the ciphertext is decrypted using their private key. The calculation process is described as follows:

Alice sends partial ciphertext to the attribute authorizer, which returns the partial decryption result $\lambda_x G - H(ID_{\mathrm{Alice}})\omega_x nG$ for each attribute $x \in \{A, B\}$. Bob sends partial ciphertext to the attribute authorizer for the partial decryption result $\lambda_C G - H(ID_{\mathrm{Bob}})\omega_C nG$. In general, an authorized user can select the vector$c \in Z_r$ to obtain the sG by calculation. When Alice and Bob have different IDs,

$$H(ID_{\mathrm{Alice}}) \neq H(ID_{\mathrm{Bob}}).$$

Then there is

$$\sum_{x \in \{\mathrm{A,B,C}\}} c(\lambda_x G - \omega_x(ID)nG) \neq sG.$$

Therefore, the original ciphertext cannot be decrypted, and the collusion attack is invalid.

6 Solution analysis and performance analysis

This paper uses the cloud storage platform of the Alibaba Cloud (equipped with a CentOS 7.3 64-bit system, 4-core CPU with a main frequency of 2.5GHz, 16GB memory, intranet broadband of 0.8 Gbps, a public network broadband of 100 Mbps, a system disk with a high efficiency cloud disk of 40 GB) to store ciphertext. The remaining components are deployed on the physical host of the lab. The specific hardware configuration is Inter(R) Core (TM) i5–2400 3.10 GHz; the memory is 2 GB; the operating system is the Ubantu 14.04.5 LTS server; the kernel version number is 4.4.0–31-generic; and the maximum bandwidth is 100Mbit/s. The 160 b elliptic curve group based on the super-singular curve $y^2 =$

$x^3 + x$ on the 512b finite field is employed; the encryption algorithm uses the algorithm library PBC; the test program is written in C language; and the length of the key is 160 bits.

The ESAD scheme proposed in this paper is compared with several assured deletion schemes. The comparison considers the scheme functionality, communication costs, and computing performance. To illustrate the functional advantages of the ESAD scheme, it is compared with the ISS scheme [10] and the ADBST scheme [11], which is based on the key management and the CP-ABE-R scheme [13] and the AD-KP-ABE scheme [15], which is based on the access control strategy. The remainder of the comparison is only compared with the CP-ABE-R scheme and the AD-KP-ABE scheme of the same type as the ESAD scheme. The descriptors are described as follows: $|C_1|$ indicates the length of the data elements in G; $|C_p|$ indicates the length of the data elements in Z_p; $|C_M|$ indicates the length of the plaintext; $|C_T|$ indicates the length of the elements in the G_T; t indicates the number of attributes in the system; k indicates the number of attributes of the user key; n_{attr} represents the total number of all user attributes; n_{user} represents the total number of users in the entire system; n_{ra} represents the number of attributes revoked in the system; n_{col} represents the number of columns in the ciphertext access structure; and n_{str} represents the total number of attributes in the access structure.

6.1 Function description

Functional comparisons are primarily compared in terms of security and fine-grained access control. As shown in Table 2, the ESAD scheme implements user revocation at the attribute level. When an attribute of users is revoked, it does not affect the normal access of other legitimate attributes of the users. The ESAD scheme can effectively resist various attacks and prevent system failure caused by failures of a single point. Although the AD-KP-ABE scheme also implements attribute-level user revocation, it cannot resist collusion attacks and sniffing attacks compared with the ESAD scheme, and cannot prevent failures of a single point. Thus, the security is poor. The CP-ABE-R scheme implements system-level user revocation. Once an attribute of the user is revoked, the user loses access to all other legal attributes in the system and is inferior to the ESAD scheme and the AD-KP-ABE scheme in terms of fine-grained access control. Since the CP-ABE-R scheme applies the LSSS to process the keys, the CP-ABE-R scheme can prevent failures of a single point. The ISS scheme is highly secure and can prevent failures of a single point but does not implement fine-grained access control. The ADBST scheme does not prevent failures of a single point and does not implement fine-grained access control for ciphertext. The ESAD scheme has higher security than other schemes and has achieved fine-grained access control for ciphertext.

6.2 Communication cost

Table 3 compares the communication costs of the ESAD scheme with costs of the AD-KP-ABE scheme and the CP-ABE-R scheme. Communication costs are primarily generated by keys and ciphertext. The communication between AKMS and DO is attributed to the key and the revocation request. For the ESAD scheme, since the least public key is employed, and the deletion operation only sends the attribute to be deleted, the communication overhead is lower than that of the other two schemes. The communication between AKMS and AU is primarily generated by a private key. However, in the ESAD scheme, the private key is stored in the attribute authorizer. When the authorized user decrypts the ciphertext, only the ciphertext content related to the user attribute needs to be sent to the authorized user for partial decryption, and then the user is authorized to return the result. Thus, the communication overhead is very low. The remaining two schemes need to send all private keys; thus, the communication overhead is large. The communication overhead between CSP and DO and AU is primarily related to ciphertext. In the AD-KP-ABE scheme, the CSP is responsible for the ciphertext update and then returns the verification result; thus, the communication overhead of CSP and DO in the AD-KP-ABE scheme is larger than the other two schemes. In terms of communication overhead between CSP and AU, the communication overhead of the three schemes are similar. In general, the ESAD solution is better than the other two solutions in terms of communication overhead.

6.3 Calculation performance

In this paper, the three schemes are compared in terms of key generation time, encryption time, decryption time and data

Table 2 Function comparison

Scheme	ESAD	AD-KP-ABE	CP-ABE-R	ISS	ADBST
Collusion attack	Yes	No	No	–	–
VLAN hopping	Yes	Yes	Yes	Yes	Yes
Sniffing attack	Yes	No	No	Yes	Yes
Single point of failure	Yes	No	Yes	Yes	No
Fine-grained access control	Attribute level	Attribute level	System level	No	No

Table 3 Comparison of communication costs

Scheme	ESAD	AD-KP-ABE	CP-ABE-R
AKMS&DO	$t\|C_1\| + (n_{\text{ra}} + 1)\|C_p\|$	$(n_{\text{attr}} + 2)\|C_1\| + \|C_T\| + (n_{\text{attr}} + n_{\text{ra}} + 1)\|C_p\|$	$(n_{\text{col}} \cdot t + 6)\|C_1\| + \|C_T\| + \|C_p\|$
AKMS&AU	$2k\|C_1\|$	$\|C_1\| + (n_{\text{str}} + 2)\|C_p\| + \|C_T\|$	$(k + 3 + n_{\text{col}})(\text{lb}n_{\text{user}} + 1)\|C_1\|$ $+(n_{\text{col}} \cdot n_{\text{str}} + 1)\|C_T\|$
CSP&DO	$(n_{col} \cdot n_{\text{str}})\|C_p\| + \|C_M\| + 2n_{\text{str}}\|C_1\|$	$\|C_M\| + (n_{\text{attr}} + 1)\|C_1\| + (n_{\text{attr}} + n_{\text{ra}} + 1)\|C_p\|$	$3\|C_1\| + (n_{\text{col}} \cdot n_{\text{str}} + 1)\|C_T\| + \|C_M\|$
CSP&AU	$(n_{col} \cdot n_{\text{str}})\|C_p\| + \|C_M\| + 2n_{\text{str}}\|C_1\|$	$\|C_M\| + (n_{\text{attr}} + 1)\|C_1\| + n_{\text{str}}\|C_p\|$	$3\|C_1\| + (n_{\text{col}} \cdot n_{\text{str}} + 1)\|C_T\| + \|C_M\|$

deletion time. The data file with size 1 M is employed, and $n_{\text{col}} = 4$, $n_{\text{ra}} = 1$ and $n_{\text{user}} = 10$ are employed [10].

As shown in Fig. 4, the key generation time and the number of user attributes linearly increase. The key generation time of the ESAD scheme is the shortest time, whereas the key generation time of the CP-ABE-R scheme is the longest time. As the key generation of the ESAD scheme is only related to the number of user attributes, the AD-KP-ABE scheme is related to the number of attributes in the entire access structure, and the key generation time of the CP-ABE-R scheme is related to not only the number of the user attributes but also the ciphertext access structure column vector n_{col} and the number n_{user} of users, the key generation time of the CP-ABE-R scheme is considerably higher than the other two schemes.

As shown in Fig. 5, the encryption time increases linearly with the number of attributes in the access structure. The encryption time of the ESAD scheme is the shortest, the encryption time of the AD-KP-ABE scheme is slightly higher than that of the ESAD scheme, and the encryption time of the CP-ABE-R scheme is much larger than other schemes. The file encryption time of the CP-ABE-R scheme is related to not only the number of attributes in the access structure but also the column vector n_{col} of the access structure matrix. Although the ESAD scheme is also related to the two aspects, the encryption operation of ESAD is only a simple scalar multiplication operation on the access control matrix. However, the CP-ABE-R scheme is to operate on the bilinear pair, which causes a large time overhead. Thus, the encryption time of the CP-ABE-R scheme is considerably larger than the encryption time of other schemes.

As shown in Fig. 6. The decryption times of the ESAD scheme and the AD-KP-ABE scheme and the CP-ABE-R scheme increase with an increase in the number of decryption attributes. However, the ESAD scheme significantly reduces the decryption time of the users. As the number of attributes increases, the decryption time overhead difference of the three schemes gradually increases. That is, when the number of attributes is sufficiently large, the decryption time of the ESAD scheme is significantly better than the decryption time of the other two schemes.

Figure 7 shows the relationship between the data deletion time and the number of attributes. When deleting data, the key or ciphertext needs to be updated. The ESAD scheme has the shortest time expenditure among the three schemes. The data deletion time of the three schemes does not change with the number of attributes, and is stable within a certain range of values. The ESAD scheme and the CP-ABE-R scheme primarily update the key; however, the AD-KP-ABE scheme primarily updates the ciphertext. The data deletion efficiency of the ESAD scheme and the AD-KP-ABE scheme is significantly better than the CP-ABE-R scheme. However, the ESAD scheme and the AD-KP-ABE scheme need to verify the deletion results of the data after deleting the data, which increases the time overhead of the two schemes. When

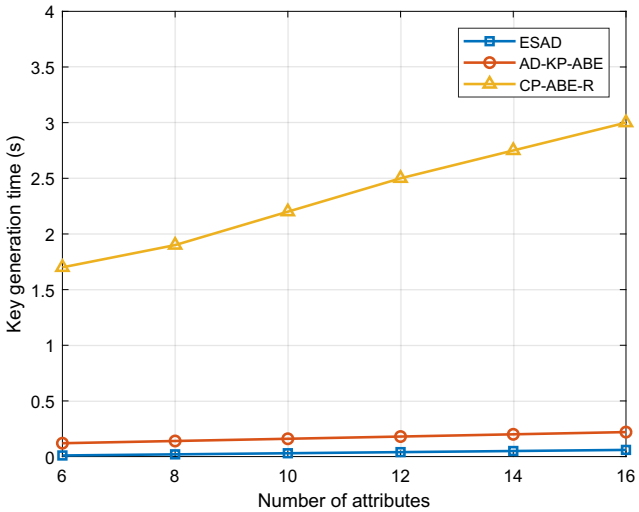


Fig. 4 Key generation time

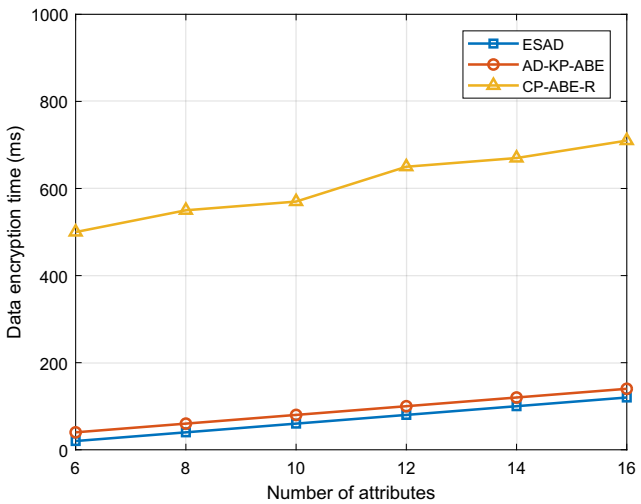


Fig. 5 Data encryption time

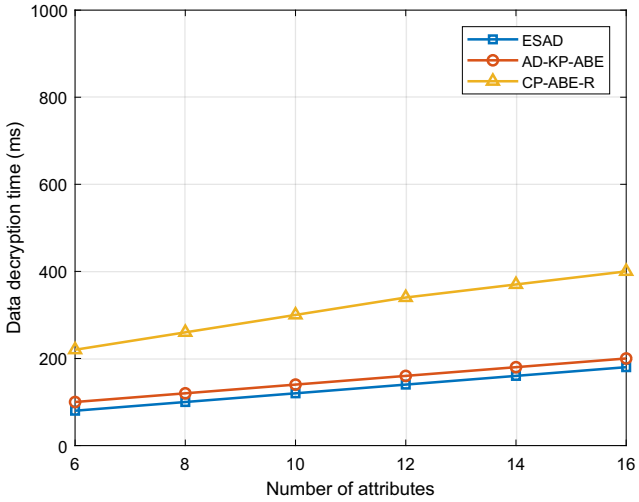


Fig. 6 Data decryption time

deleting data, the three schemes update and delete the attributes that need to be deleted, and the time consumption of deleting data does not increase as the number of attributes increases.

7 Conclusion

This paper proposes an efficient cloud data assured deletion scheme that enables users to encrypt and decrypt and delete original data in a short period. In this scheme, if the attribute authorizer changes the attribute value of the maintained user attribute list, and the data owner does not update the public key, the user whose attribute is revoked cannot decrypt the ciphertext of the cloud, only the attribute satisfies the access policy and users who have not been revoked can successfully decrypt the ciphertext. Thus, fine-grained access and deletion of cloud data are achieved. Based on the DDH hypothesis, this paper makes a security proof of the selected access structure plaintext attack using the scheme in the standard model. The performance analysis and experimental verification of the scheme are performed. The experimental results show that compared with the existing schemes, the scheme has distinct advantages in terms of system function and time overhead. With an increase in the encrypted files, the time advantage will be more distinct. The scheme also realizes the reasonable allocation of the work in the system, which reduces the computational overhead of a single party and increases the communication overhead. However, this allocation can be disregarded as only a small amount of information is required to be transmitted. However, when the number of users of the ESAD scheme increases, the attributes list maintained by the attribute authorizer will also increase, which will increase the time overhead of data deletion and verification. Solving the problem that deletion and verification efficiency decrease with an increase in the number of users needs to be addressed in future studies.

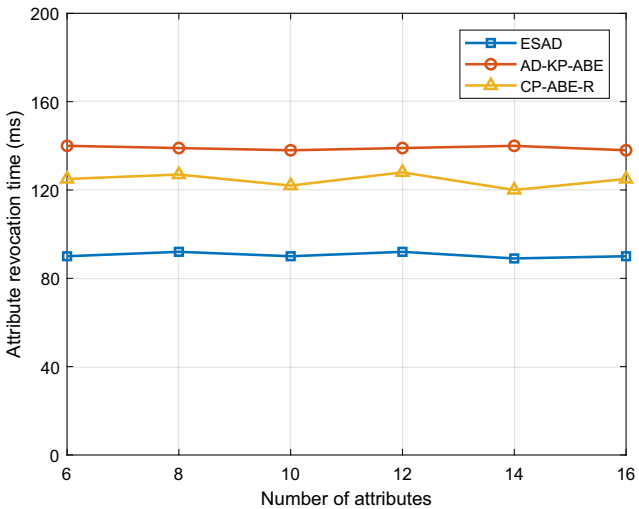


Fig. 7 Attribute revocation time

References

1. Reardon J, Basin DA, Capkun S (2016) Secure data deletion[J]. Information Security and Cryptography (IS&C)
2. Xiong JB, Li FH, Wang YC et al (2016) Research progress on cloud data assured deletion based on cryptography[J]. J Commun 37(8): 167–184
3. Liu ZL, Li T, Li P et al (2018) Verifiable searchable encryption with aggregate keys for data sharing system[J]. Futur Gener Comput Syst 78(2):778–788
4. Liu ZL, Huang YY, Li J et al (2018) DivORAM: towards a practical oblivious RAM with variable block size[J]. Inf Sci 447
5. Li T, Liu ZL, Li J et al (2017) CDPS: a cryptographic data publishing system[J]. J Comput Syst Sci 89:80–91
6. Agrawal S, Mohassel P, Mukherjee P et al (2018) DiSE: distributed symmetric-key encryption[C]. ACM Conference on Computer and Communications Security(CCS):1993–2010
7. Li H, Sun WH, Li FH et al (2014) Secure and privacy-preserving data storage Service in Public Cloud[J]. Journal of Computer Research and Development 51(7):1397–1409
8. Perlman R. File system design with assured delete[C]. Third IEEE International Security in Storage Workshop (SISW). San Francisco, CA, USA, c2005:83–88
9. Perlman R. File system design with assured delete[C]. The 14th Annual Network and Distributed System Security (ISOC NDSS). San Diego, USA, c2007:1–7

10. Xiong JB, Yao ZQ, Ma JF et al (2014) A secure self-destruction scheme with IBE for the internet content privacy[J]. Chinese Journal of Computers 37(1):139–150
11. Yao W, Chen Y, Wang D (2017) Cloud multimedia file assured deletion based on bit stream transformation with chaos sequence[C]. International Conference on Algorithms and Architectures for Parallel Processing. Springer, Cham, p 441–451
12. Zhang K, Yang C, Ma JF et al (2016) Novel cloud data assured deletion approach based on ciphertext sample slice[J]. J Commun 36(11):108–117
13. Liang X, Lu R, Lin X, Liang X, Lu R, Lin X et al (2010) Ciphertext policy attribute based encryption with efficient revocation[J]. IEEE Symposium on Security and Privacy(S&P) 2008:321–334
14. Wang GB, Liu HT, Wang CL et al (2018) Revocable attribute based encryption in cloud storage[J]. Journal of Computer Research and Development 55(6):1190–1200
15. Xue L, Yu Y, Li Y et al (2018) Efficient attribute-based encryption with attribute revocation for assured data deletion[J]. Inf Sci:1–11
16. Zhao ZY, Zhu ZQ, Wang JH et al (2018) Revocable attribute-based encryption with escrow-free in cloud storage[J]. J Electron Inf Technol 40(1):1–10
17. Li B, Huang DJ, Wang ZJ et al (2018) Attribute-based access control for ICN naming scheme[J]. IEEE Transactions on Dependable and Secure Computing(TDSC) 15(2):194–206
18. Robshaw M, Yin Y (1997) Elliptic curve cryptosystems. An RSA Laboratories Technical Note Revised June 27
19. Xu QL, Li DX (1999) Elliptic curve cryptosystems[J]. Journal of Computer Research and Development 36(11):1281–1288
20. Peng Q, Tian YL (2017) A secret sharing scheme based on multilinear Diffie-Hellman problem[J]. Acta Electron Sin 45(1): 200–205
21. Brent Waters (2011) Ciphertext policy attribute based on encryption: An expressive, efficient, and provably secure realization. Proceedings of the 14th international conference on practice and theory in public key cryptography, Taormina, Italy, p 53–70
22. Lewko A, Okamoto T, Sahai A et al (2010) Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption[C]. Springer, Berlin, pp 62–91