FDSCD: Fast Deletion Scheme of Cloud Data

Tong Shao[1](✉), Yuechi Tian[2], Zhen Li[1,2], and Xuan Jing[1]

[1] Cyberspace Security and Computer College, Hebei University, Baoding 071002, China
15613668123@163.com
[2] School of Computer Science and Technology, HUST, Wuhan 430074, China

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 the Fast deletion scheme of cloud data. 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.

To protect the confidentiality and privacy of data, cloud data must be encrypted before they are outsourced to the cloud [1–5]. 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.

J. Li et al. (Eds.): SPNCE 2019, LNICST 284, pp. 354–361, 2019.
https://doi.org/10.1007/978-3-030-21373-2_27

Assured deletion schemes based on key management. proposed a cloud data security self-destruction scheme (ISS) that is based on identity-based encryption (IBE) [8], which divides the data to be protected into different security levels according to different sensitivity levels. Yao et al. [9] proposed a cloud data assured deletion scheme that is based on bitstream conversion.

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. For fine-grained access control problems, Liang et al. [10] 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. Wang et al. [11] proposed an assured deletion scheme that is based on attribute revocation, which is a CP-ABE scheme that supports attribute-level user revocation. Xue et al. [12] proposed a cloud data assured deletion scheme that is based on hash tree verification for attribute revocation.

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, the Fast deletion scheme of cloud data (FDSCD) 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 [13], 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.

2 System Model

2.1 System Model

The system model of the FDSCD solution is shown in Fig. 1. The model includes four types of entities: Attribute Key Management System (AKMS), Data Owner (DO), Cloud Service Provider (CSP), Authorized user (AU).

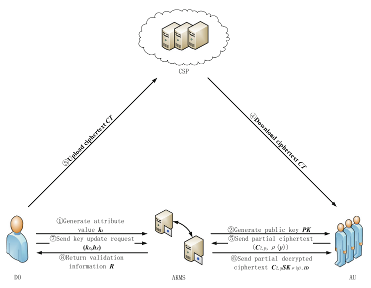
CSP
③Upload ciphertext CT
④Download ciphertext CT
①Generate attribute value k_i
⑦Send key update request (k_x,h_x)
⑧Return validation information R
②Generate public key PK
⑤Send partial ciphertext $(C_{2,y}, \rho(y))$
⑥Send partial decrypted ciphertext $C_{2,y}SK_{\rho(y),ID}$
DO
AKMS
AU

Fig. 1. The system model

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. 2. 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.

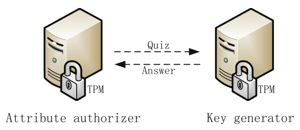

Fig. 2. Internal structure of attribute key management system

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

3 Detailed Design of the FDSCD Scheme

The workflow of the FDSCD scheme is shown in Fig. 1. 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: $\{0,1\}^* \to Z_r^*$ is randomly selected: the user ID is mapped to the integer domain Z_r.

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

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.

$$MSK = (k_1, k_2, \cdots, k_S, n, G),$$

Generate the system public key parameters.

$$PK = (k_1G, k_2G, \cdots, k_SG).$$

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

Table 1. User attribute list

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

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:

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

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 [14, 15] as follows:

Define the access control strategy ($\boldsymbol{A}$, ρ), 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 [16], 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 $\boldsymbol{v} = (s, v_2, v_3, \cdots, v_l)^{\mathrm{T}}$, and calculate the inner product $\lambda_x = \boldsymbol{A}_x \cdot \boldsymbol{v}$ 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 $\boldsymbol{u} = (0, u_2, u_3, \cdots, u_l)^{\mathrm{T}}$, and calculate the inner product $\boldsymbol{\omega}_x = \boldsymbol{A}_x \cdot \boldsymbol{u}$ for each row $\boldsymbol{A}_x$ of the matrix $\boldsymbol{A}$.

The data owner outputs the ciphertext

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

Step 4. Data decryption

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

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 ($C_{2,y}$, $\rho(y)$)

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

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

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

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

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

The authorized user calculates

$$C - sG = M.$$

The original data M are obtained.

Step 5. 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 $x \in S_{ID_{RL_x}}$.

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

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

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

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

Step 6. 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 $\tilde{X}_{ID}$ 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 $\tilde{X}_{RL_x}$ of the user set RL_x. Calculate the local hash tree root value $\tilde{R}$ using the hash value $\tilde{X}_{RL_x}$ and the auxiliary authentication information Ω_{RL_x}. If $\tilde{R} = R$, the attribute update operation requested by the data owner is successful; otherwise, the request execution fails.

4 Conclusion

This paper proposes the fast deletion scheme of cloud data 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. 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.

References

1. Reardon, J., Basin, D.A., Capkun, S.: Secure data deletion. Inf. Secur. Cryptogr. (2016)
2. Xiong, J.B., Li, F.H., Wang, Y.C., et al.: Research progress on cloud data assured deletion based on cryptography. J. Commun. **37**(8), 167–184 (2016)
3. Liu, Z.L., Li, T., Li, P., et al.: Verifiable searchable encryption with aggregate keys for data sharing system. Futur. Gener. Comput. Syst. **78**(2), 778–788 (2018)
4. Liu, Z.L., Huang, Y.Y., Li, J., et al.: DivORAM: towards a practical oblivious RAM with variable block size. Inf. Sci. **447**, 1–11 (2018)
5. Li, T., Liu, Z.L., Li, J., et al.: CDPS: a cryptographic data publishing system. J. Comput. Syst. Sci. **89**, 80–91 (2017)
6. Agrawal, S., Mohassel, P., Mukherjee, P., et al.: DiSE: distributed symmetric-key encryption. In: ACM Conference on Computer and Communications Security, pp. 1993–2010 (2018)
7. Li, H., Sun, W.H., Li, F.H., et al.: Secure and privacy-preserving data storage service in public cloud. J. Comput. Res. Dev. **51**(7), 1397–1409 (2014)
8. Xiong, J.B., Yao, Z.Q., Ma, J.F., et al.: A secure self-destruction scheme with IBE for the internet content privacy. Chin. J. Comput. **37**(1), 139–150 (2014)
9. Yao, W., Chen, Y., Wang, D.: Cloud multimedia files assured deletion based on bit stream transformation with chaos sequence. In: Ibrahim, S., Choo, K.-K.R., Yan, Z., Pedrycz, W. (eds.) ICA3PP 2017. LNCS, vol. 10393, pp. 441–451. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-65482-9_31
10. Liang, X., Lu, R., Lin, X., et al.: Ciphertext policy attribute based encryption with efficient revocation. IEEE Symp. Secur. Priv. **2008**, 321–334 (2010)
11. Wang, G.B., Liu, H.T., Wang, C.L., et al.: Revocable attribute based encryption in cloud storage. J. Comput. Res. Dev. **55**(6), 1190–1200 (2018)
12. Xue, L., Yu, Y., Li, Y., et al.: Efficient attribute-based encryption with attribute revocation for assured data deletion. Inf. Sci., 1–11 (2018)
13. Li, B., Huang, D.J., Wang, Z.J., et al.: Attribute-based access control for ICN naming scheme. IEEE Trans. Dependable Secur. Comput. **15**(2), 194–206 (2018)
14. Peng, Q., Tian, Y.L.: A secret sharing scheme based on multilinear Diffie-Hellman problem. Acta Electron. Sin. **45**(1), 200–205 (2017)
15. Waters, B.: Ciphertext policy attribute based on encryption: an expressive, efficient, and provably secure realization. In: Proceedings of the 14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, Italy, pp. 53–70 (2011)
16. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_4