Study on Data Security Sharing Model Based on Attribute Encryption

Jindong He1,*, Zhijun Tang1, Jiwen Lian2, Fei Wu2, Zhichao Zhao1 and Xinzi Xie1

1State Grid Fujian Electric Power Research Institute, Fuzhou Fujian 350007, China
2State Grid Fujian Electric Power Company Limited, Fuzhou Fujian 350003, China

*Corresponding author e-mail: 2468537873@qq.com

Abstract. In view of the problems of key management and access control in traditional encryption technology, this paper proposes a data security sharing model based on attribute encryption. Firstly, we studied the property encryption and data sharing model. Secondly, proposed an attribute encryption algorithm. The user's private key is generated by the addition homomorphic encryption algorithm, which solves the key hosting problem. This paper proposed the data security sharing algorithm supporting partial decryption, which can reduce the calculation cost of user decryption. An efficient scheme of real-time attribute key cancellation is designed, which not only reduces the communication key, but also protects the forward and backward security of the data. Compared with the other two methods, the proposed method has lower computational and decryption costs.

1. Introduction

Cloud computing platform has abundant computing and storage resources, and concentrates a large number of users' private data. At the same time, the large-scale networked systems face more complex security threats than traditional computing systems. The characteristics of cloud computing service computing mode and multi-user sharing usage mode bring new challenges to data security [1-3].

In recent years, data security sharing model has attracted more and more attention from scholars. Literature [4] proposes a credible big data sharing model based on blockchain technology and intelligent contract to ensure the safe flow of data resources [4]. Literature [5] propose a secure data sharing scheme in OSNs based on ciphertext-policy attribute-based proxy re-encryption and secret sharing. In order to protect users' sensitive data, our scheme allows users to customize access policies of their data and then outsource encrypted data to the OSNs service provider [5]. Literature [6] proposed the big data security service model of big data providers, users and cloud service providers to realize reliable data security sharing services [6]. Although the above research has been done on data security, the existing data security sharing model based on attributes in the cloud computing platform has some shortcomings: (1) the attribute mechanism can obtain the user's private key and decrypt the data; (2) users need to run complex pairing and exponential operation when decryption; (3) when the key is withdrawn, the user's private key needs to be updated and distributed to the user. In view of the above problems, this paper proposes a secure and efficient data security sharing model based on attribute encryption.
2. Property encryption and data sharing

The rapid development of cloud computing allows users to upload data to the cloud computing platform and use the data in the cloud computing platform anytime and anywhere, but the semi-trusted environment of cloud computing also brings serious security and privacy issues [7]. Encryption technology is an effective means to protect data confidentiality. Users encrypt the data before storing it to the cloud server, and the cloud service provider cannot obtain the clear text of user data [8]. Attribute encryption is an encryption technique that supports fine-grained access control. It allows data owners to encrypt content using an access policy, and only users who have properties that satisfy the access policy can decrypt the content [9]. However, the data sharing scheme based on the implementation of attribute encryption has the problem of key hosting, that is, the attribute organization can calculate the private key of any user's attribute and decrypt the data stored in the cloud computing platform.

2.1. Property encryption

Attribute encryption takes attributes as public keys, associates the ciphertext and user private keys with attributes, and flexibly represents access policies. When the user's private key matches the access policy of the ciphertext, the user can decrypt the ciphertext. Attribute encryption includes both KP-ABE and CP-ABE. In KP-ABE, the user's attribute private key is related to the access policy, and the ciphertext is related to a set of attributes. KP-ABE algorithm mainly consists of the following four algorithms:

1. System initialization algorithm (Setup). Attribute mechanism input security parameter k, output system public key PK and system master key MK.
2. Key generation algorithm (KeyGen). Attribute mechanism inputs system master key MK and access policy T, generating private key SK for the user.
3. Encryption algorithm (Encrypt). User input system public key PK, data plaintext M and attribute set S to generate ciphertext CT.
4. Decrypt algorithm. User input ciphertext CT and private key SK. If the attribute in the ciphertext meets the access policy T, the data plaintext M is decrypted.

Since the CP-ABE ciphertext is associated with the access policy, CP-ABE is more suitable for access control in the cloud computing platform.

2.2. Data sharing model

Under the cloud computing platform, cloud service providers can upload, access, and backup and share data in the cloud computing environment through the network, providing users with convenient and fast data storage and sharing services [10]. At present, typical data sharing applications under the cloud computing platform include cloud storage platform, cloud social networking platform and cloud health platform [11]. The data sharing model under the cloud computing platform generally has three main entities: cloud service providers, data owners and users, involving various stages of creation, storage, use, sharing, archiving, key destruction in the data life cycle, as shown in figure 1 [12].
Figure 1. Data sharing model in cloud computing mode

(1) Data upload. The data owner uploads the data to the cloud computing platform and can either expose the shared data or specify that the user can access and modify the data.

(2) Data access. Authorized users can easily access data uploaded by the data owner through the cloud service provider.

(3) Data modification. Data owners can re-upload or modify data stored in cloud computing platforms. For authorized users, the data stored in the cloud computing platform can also be modified according to the usage requirements of the data.

Since the data owner's data is stored on the cloud server, the data owner does not have complete control over the data, so the user's data must be secured. One of the data security requirements under the cloud computing platform is data confidentiality, which means protecting users' data from being compromised or stolen by semi-trusted cloud server providers while ensuring that only authorized users can access the data.

3. Data security sharing model based on attribute encryption

3.1. Model conception

The model is mainly composed of five parts: attribute mechanism, cloud service provider, key server, data owner and user, as shown in figure 2.

(1) Attribute mechanism. Attribute authority is an authoritative attribute management organization responsible for distributing attributes for users, and the joint key server generates attribute private keys for users. In this scheme, we assume that the attribute mechanism is a semi-trusted third party.

(2) Key server. The key server is a semi-trusted third party, and the joint attribute mechanism generates the user's attribute private key. In addition, when the user decrypts the data, the key server performs a partial decryption operation for the authorized user, and then returns the partially decrypted ciphertext to the user.

(3) Cloud service providers. Cloud service providers provide data storage and download services in the form of networks through dynamic resources. Cloud service providers are also semi-trusted third parties. The owner of the stored data uploads the encrypted data and tries to get clear text.

(4) Data owner. The data owner first encrypts the data using the data key, then encrypts the data key using the access policy, and finally uploads the ciphertext to the cloud service provider.

(5) Users. When a user downloads data from a cloud service provider, the key server decrypts a portion of the user's encrypted message if the user's properties meet the encrypted data access policy.
3.2. Algorithm definition
CP-ABE algorithm proposed by Bethencourt et al., the model is as follows [13]:
(1) The Setup (k). Input security parameter k, property agencies generate public key PKA and
private key SKA, and servers also generate public key PKK and private key SKK.
(2) The KeyGen (). Based on the addition homomorphic encryption algorithm, the property agency
joint key server generates the user key SKA and sends it to the user safely.
(3) KeyGen. Key server input security parameter r and user attribute set S to generate attribute key
SKK.
(4) Encrypt (PKA, PKK, M, T). Data owners input the public key PKA of the property organization,
the public key PKK of the key server, data plaintext M, access strategy tree T, and output the encrypted
data CT. Firstly, based on symmetric encryption algorithm, data plaintext M is encrypted with random
data keys, and then DK is encrypted with access strategy tree T based on CP-ABE algorithm.
(5) PartDec (CT, SKK). The key server enters the ciphertext CT and the users attribute key SKK. If
the user's attributes meet the ciphertext CT access policy, the output part of the ciphertext CTp
decryption.
(6) Decrypt (CTp, SKA). User input partially decrypted ciphertext CTP, user key SKA. Decrypt
DK, and then use DK to decrypt data plaintext M.
(7) ReKey (SKK, a i). When the attribute authority revokes a user's attribute ai, the key server
regenerates the attribute key SKK that has the attribute ai.
(8) ReEncrypt (CT, a i). When the attribute agency revokes a user's attribute a i, the key server
reencrypts the ciphertext CT related to the attribute a_i, and outputs the heavily encrypted ciphertext
CT'.

3.3. Algorithm description
(1) System initialization
The attribute mechanism runs the Setup algorithm to construct A order of p bilinear group G1,
where the generation element of G1 is denoted by g, and the corresponding bilinear mapping is $e^\wedge$:
$G_1 \times G_1 \rightarrow G_2$, and defined H: \{0,1\} $\rightarrow$ $G_1$. The attribute mechanism randomly selects $\alpha \in \mathbb{Z}_p$ and
generates the public key and private key as follows:
$$PK_a = e(g,g)^\alpha, MK_a = g^\alpha$$

Then, the key server selects $\beta \in \mathbb{Z}_p$ randomly and generates the public and private keys as follows:
$$PK_k = g^\beta, MK_k = \beta$$
(2) Key generation
The attribute mechanism and key server first run the KeyGen algorithm and then generate the user key based on the addition homomorphic encryption algorithm.

(a) Attribute mechanism generates \( W = Enc(\alpha) \) and sends it to the key server.

(b) The key server selects \( \gamma \in \mathbb{Z}_p \), Generate \( V = (W \oplus Enc(\gamma)) \otimes \beta \) and send it to the property organization.

(c) Attribute mechanism decrypts \( V \) to get \( X \) based on addition homomorphic encryption algorithm.

\[
X = Dec(MK_{\alpha}, V) = (\alpha + \gamma)\beta
\]  

(d) Attribute mechanism random selection \( \tau \in \mathbb{Z}_p \), and computing \( Z = g^{X/\tau} = g^{(\alpha+\gamma)\beta/\tau} \) then sent to the key server

(e) The key server computes \( N = Z^{1/\beta^2} = g^{(\alpha+\gamma)/\beta^2} \) and sends it to the property organization.

(f) The property organization uses \( \tau \) to generate the user key \( SK_{\alpha} = D = N^{\tau} = g^{(\alpha+\gamma)/\beta} \), and send it to the user safely.

(g) The key server then runs the KeyGen algorithm, randomly selects \( \gamma_j \cup \mathbb{Z}_p \) each attribute for the user \( \alpha_j \in S \), and generates and stores the attribute key \( SK_k \). The user’s attribute private key \( SK \) is composed of user key \( SK_{A} \) and attribute key \( SK_{K} \).

\[
SK_k = ((D_j = g^{\gamma_j}H(j)^{\gamma_j}, D'_j = g^{\gamma_j/\gamma_{j\in S}}))
\]  

(3) Data encryption
The data owner runs the Encrypt algorithm, sets the access strategy tree \( T \), encrypts the data \( M \), and outputs the ciphertext \( CT \). First, the data owner selects \( DK \in \mathbb{Z}_p \) randomly and uses \( DK \) to encrypt the data \( M \) based on the symmetric encryption algorithm. Then, construct the access policy tree \( T \), define a degree polynomial \( q \) for each node \( x \) in the tree in a top-down manner, and select \( s \in \mathbb{Z}_p \) at random. For the root node \( R \) of tree \( T \), define \( q_R(0) = s \). For other node \( x \) of tree \( T \), define \( q_x(0) = q_{parent(x)}(index(x)) \), and select random parameters to complete the definition of \( q_x \).

Suppose \( Y \) represents the set of attributes corresponding to leaf nodes in the access strategy tree \( T \), and the ciphertext is constructed as follows:

\[
CT = (T, E = SEnc_{DK}(M), \tilde{C} = DK : \hat{e}(g, g)^{as}, C = g^{\beta S}, \\
\{C_y = g^{q_y(0)}, C'_y = H(attr_y)^{q_y(0)}\}_{y \in Y})
\]  

The data owner uploads the ciphertext to the cloud service provider.

(4) Data decryption
After the user gets the ciphertext from the cloud service provider, sends the decryption request to the key server. The key server runs the PartDec algorithm and decrypts the ciphertext using the attribute key section. The process of decryption is implemented by recursive algorithm, which defines the recursive algorithm DecryptNode (CT, SKK, x), input ciphertext CT, attribute key SKK and node x in the access strategy tree T. If x is a leaf node, define i=attrx.

If \( i \in S \), then:
\[
\text{DecryptNode}(CT, SK_K, x) = \frac{e(D_1, C_j)}{e(D_1, C_j')} = \frac{e(g^{\gamma H(i)^{x_j}}, g^{q_j(0)})}{e(g^{\gamma H(i)}, g^{q_j(0)})} = e(g, g)^{q_j(0)}
\] (6)

If \( x \) is not a leaf node, run the DecryptNode algorithm until the root node: \( z \) of the child node of \( c \), run the DecryptNode \( (CT, SK_K, x) \) algorithm, and the results are saved in \( F_z \). Let \( S_\epsilon \) be the set of any \( k \), node \( z \), and it satisfies the DecryptNode \( (CT, SK_K, x) \). The calculation is as follows:

\[
F_x = \prod_{z \in S_\epsilon} F_x^\Delta_{i,S_{\epsilon}(t)} = e(g, g)^{q_z(0)}
\] (7)

\( \Delta_{i,S_{\epsilon}(t)} \) is the Lagrangian coefficient. If the user's attribute \( S \) meets the ciphertext access policy tree \( T \), the cloud service provider computes as follows:

\[
A = \text{DecryptNode}(CT, SK_K, R) = e(g, g)^{x_j}
\] (8)

The cloud service provider sends \( CT_p = (E, \tilde{C}, C, A) \) partially decrypted ciphertext \( A \) to the user. Users run the Decrypt algorithm and use the user key \( SK_R \) to Decrypt the DK as follows:

\[
\tilde{C} / (e(C, D) / A) = DK \cdot e(g, g)^{x_j} / (e(g^{\beta(j)}, g^{q_j(0) + x_j}) / e(g, g)^{x_j}) = DK
\] (9)

Finally, the user USES DK to decrypt data plaintext M:

\[
M = \text{SDec}_{DK}(E)
\] (10)

The key server then runs the ReEncrypt algorithm to reencrypt the ciphertext, randomly select \( s \in Z_p \) and update all ciphertext containing the attribute \( a_i \) as follows:

\[
CT' = (T, E = SEnc_{DK}(M), \tilde{C} = DK \cdot e(g, g)^{q_j(0) + x_j}), \quad C_i = g^{q_j(0) + x_j}, \quad C_i' = (H(\text{attr})^{q_j(0) + x_j})^{1/\delta}, \\
\{C_y = g^{q_j(0) + x_j}, C_y' = H(\text{attr})^{q_j(0) + x_j} \}_{y \in \mathbb{F}}
\] (11)

Key refers to the ciphertext that the server reencrypts after the revocation of the unrevocation user can still run the PartDec and the Decrypt algorithm to Decrypt the reencrypting ciphertext \( CT' \). If the attributes of the unrevoked user satisfy the access policy tree \( T \), the calculation is as follows:

\[
A = \text{DecryptNode}(CT', SK_K, R) = e(g, g)^{q_j(0) + x_j}
\] (12)

The user then decrypts the DK without revoking it as shown in equation (13). Finally, the user USES DK to decrypt the data plaintext M.

\[
\tilde{C}' / (e(C', D) / A) = DK \cdot e(g, g)^{q_j(0) + x_j} / (e(g^{\beta(j)}, g^{q_j(0) + x_j}) / e(g, g)^{x_j}) = DK
\] (13)
3.4. Safety analysis

(1) Data confidentiality

First, under the difficult assumption of DBDH, the data security sharing model based on attribute encryption is IND-CPA security. The data owner encrypts the data based on the CP-ABE algorithm before uploading the ciphertext CT to the cloud service provider. Since the symmetric key is a random number, if the user's attribute cannot meet the access policy T of the ciphertext, \( e(g, g)^{\gamma x} \) cannot be restored and DK can be decrypted. The key server helps the user to partially decrypt the ciphertext and cannot decrypt the DK without the user key SKA.

In addition, the attribute mechanism and key server generate the user's attribute private key based on the addition homomorphic encryption algorithm. The property organization cannot know \( \beta \) in the user's key, and the key server does not know \( \alpha \) in the user's key. Therefore, neither the attribute mechanism nor the key server can generate the user key separately.

(2) Resist collusion attacks

Through the decryption process, it can be known that the user can decrypt \( e(g, g)^{\gamma x} \) only if the set of properties that he owns satisfies the access policy. The random and unique \( \gamma \) is used by the attribute mechanism to generate the attribute private key for each user, ensuring that the attribute private key of each user is different. Even if the collusive user can calculate \( e(g, g)^{\gamma x(0)} \) of the corresponding node x, but cannot calculate \( e(g, g)^{\gamma x} \), and therefore cannot decrypt DK.

(3) Forward and backward security

When the user revokes the attribute, the cloud service provider USES \( s' \) to reencrypt the ciphertext related to the attribute. The attribute institution uses \( \delta' \) to update the attribute key corresponding to the user, which ensures the forward security of the data. When the new user owns the attribute, the attribute institution updates the attribute key of the user who also owns the attribute. Meanwhile, the cloud service provider USES \( s' \) to rescript the ciphertext related to the attribute, which ensures the backward security of the data.

4. Performance comparison

4.1. Experimental comparison

The proposed model is compared with other models in the calculation and communication performance, and the results are shown in table 1. Where, S represents the number of attributes of the user, N represents the number of attributes in the access policy, R represents the number of users who have not revoked attributes, C1 represents the size of elements within G1, C2 represents the size of elements within Zp, P represents the pairing operation, E1 represents the exponential operation within G1 and E2 represents the exponential operation within G2, and other operations can be ignored.

Through experimental calculation and analysis of the model in this section when users decrypt data calculation overhead. The experimental environment is 32-bit Ubuntu12 system, the main frequency is 2.53ghz, the memory is 2GB, and the open source PBC library is used to implement attribute encryption algorithm PBC library is built on the well-known open source mathematical operation library GMP, providing a variety of operations based on bilinear pairing. In the experiment, the number of attributes owned by the user is set as 100, and the calculation time of the model in this section and the model of Hur et al. at the time of user data decryption is calculated for different number of attributes contained in the access strategy, as shown in figure 3. The model in this paper is compared with the existing cloud computing platform data security sharing model in terms of key hosting, partial decryption, forward security, and backward security, and the results are shown in table 2.
Table 1. Performance comparison

| Method          | Structure | Data encryption | Data decryption | The key to update |
|-----------------|-----------|-----------------|-----------------|-------------------|
| Hur's method [14] | tree      | (2N+1)E₁+E₂    | (S+2N+2)P+2SE₁+NE₂ | RC₁              |
| Yang's method [15] | LSSS      | (3N+1)E₁+E₂    | E₂              | RC₁              |
| this paper's method | tree      | (2N+1)E₁+E₂    | P               | 0                |

Figure 3. Comparison of decryption time

Table 2. Comprehensive comparison

| Method          | Key escrow   | Part of the decryption | Forward security | Safety after |
|-----------------|--------------|-------------------------|------------------|--------------|
| Hur's method [14] | two-way security calculation | No                      | Yes              | Yes          |
| Yang's method [15] | No           | Yes                     | Yes              | Yes          |
| this paper's method | add homomorphic encryption | Yes                    | Yes              | Yes          |

4.2. Results analysis

The comparison results show that in the model in this section, the user has obvious advantages in data decryption, only one pairing operation is needed, and its calculation cost is independent of the number of attributes in the access strategy. In addition, the model in this section does not require the user to update the property key when the property is revoked, while all other models require additional communication overhead to update the user's property key.

Both the models in this section and those of Hur et al have solved the key escrow problem, which can prevent the property agency from using the generated property private key to decrypt the data owner's data. In addition, both the model in this section and the model of Yang et al. allow the key server to partially decrypt the ciphertext, reducing the user's decryption overhead. In addition, the model in this section also implements forward and backward security.

5. Conclusion

In this paper, a data security sharing model based on attribute encryption is proposed, and the user's attribute private key is generated by the addition homomorphic encryption algorithm to solve the key hosting problem. At the same time, the model not only supports partial decryption of the key server, reduces the computational overhead of user decryption, but also realizes efficient real-time attribute key removal. This paper realizes: (1) Using the addition homomorphic encryption algorithm to generate the user's attribute private key, solving the key hosting problem. (2) The data security sharing algorithm supporting partial decryption is proposed, which can reduce the computational overhead of user decryption. (3) An efficient scheme of instant attribute key withdrawal is designed, which has less communication key and protects the forward and backward security of data.
References

[1] Qiang Tang. Nothing is for Free: Security in Searching Shared and Encrypted Data [J]. IEEE Transactions on Information Forensics and Security, 2014, 9 (11): 1943-1952.

[2] Lei Zhao, Xiangyu Ren, Mengleng Liu, et al. Collaborative reversing of input formats and program data structures for security applications [J]. 2014, 11 (9): 135-147.

[3] Yang Mengke, Zhou Xiaoguang, Zeng Jianqiu, et al. Challenges and solutions of information security issues in the age of big data [J]. China Communications, 2016, 13 (3): 193-202.

[4] Li Yue, Huang Junqin, Qin Shengzhi, et al. Big Data Model of Security Sharing Based on Blockchain [C]. 2017 3rd International Conference on Big Data Computing and Communications (BIGCOM), Chengdu, China, 10-11 Aug.2017, 117 – 121.

[5] Huang Qinlong, Ma Zhaofeng, Yang Yixian. Improving security and efficiency for encrypted data sharing in online social networks [J]. China Communications, 2014, 11 (3): 104-117.

[6] Hai-ting Cui. Research on the model of big data serve security in cloud environment [C]. 2016 First IEEE International Conference on Computer Communication and the Internet (ICCCI), 13-15 Oct. 2016, Wuhan, China.

[7] Hui Ma, Rui Zhang, Wei Yuan. Comments on “Control Cloud Data Access Privilege and Anonymity With Fully Anonymous Attribute-Based Encryption” [J]. IEEE Transactions on Information Forensics and Security, 2016, 11 (4): 866-867.

[8] Junbeom Hur. Improving Security and Efficiency in Attribute-Based Data Sharing [J]. IEEE Transactions on Knowledge and Data Engineering, 2013, 25 (10): 2271-2282.

[9] Jianan Hong, Kaiping Xue, Wei Li. Comments on “DAC-MACS: Effective Data Access Control for Multiauthority Cloud Storage Systems”/Security Analysis of Attribute Revocation in Multiauthority Data Access Control for Cloud Storage Systems [J]. IEEE Transactions on Information Forensics and Security, 2015, 10 (6): 1315-1317.

[10] Huang Qinlong, Ma Zhaofeng, Yang Yixian, et al. Improving security and efficiency for encrypted data sharing in online social networks [J]. China Communications, 2014, 11 (3): 104-117.

[11] Kyle Chard, Steven Tuecke, Ian Foster. Efficient and Secure Transfer, Synchronization, and Sharing of Big Data [J]. IEEE Cloud Computing, 2014, 1 (3): 46-55.

[12] Xuefeng Liu, Yuqing Zhang, Boyang Wang, et al. Mona: Secure Multi-Owner Data Sharing for Dynamic Groups in the Cloud [J]. IEEE Transactions on Parallel and Distributed Systems, 2013, 24 (6): 1182-1191.

[13] Bethencourt J, Sahai A, Waters B. Ciphertext-Policy Attribute-Based Encryption [J]. In Proceedings of 2007 IEEE Symposium on Security and Privacy, IEEE, 2007, 321-334.

[14] Hur J. Improving Security and Efficiency in Attribute-Based Data Sharing, IEEE Transactions on Knowledge and Data Engineering, 25 (10), 2013, 2271-2282.

[15] Yang Y, Zhang Y. A Generic Scheme for Secure Data Sharing in Cloud. In Proceedings of the 2011 40th International Conference on Parallel Processing Workshops (ICPPW), 2011, 145-153.