Security Analyses of a Data Collaboration Scheme with Hierarchical Attribute-based Encryption in Cloud Computing

Wei-Liang Tai[1], Ya-Fen Chang[2], and Wen-Hsin Huang[2]

(Corresponding author: Ya-Fen Chang)

Department of Information Communications, Chinese Culture University[1]

No. 55, Hwa-Kang Road, Taipei, Taiwan

Department of Computer Science and Information Engineering, National Taichung University of Science and Technology[2]

No. 129, Section 3, Sanmin Road, Taichung, Taiwan

(Email: cyf@nutc.edu.tw)

(Received Oct. 16, 2018; Revised and Accepted Feb. 7, 2019; First Online June 25, 2019)

Abstract

With the prevalence of cloud computing, users store and share confidential data in the cloud while this approach makes data security become an important and tough issue. To ensure data security, cloud service providers must provide efficient and feasible mechanisms to provide a reliable encryption method and a suitable access control system. In order to realize this ideal, Huang *et al.* proposed a data collaboration scheme with hierarchical attribute-based encryption. After analyzing Huang *et al.*'s scheme, we find that one weakness exists in their scheme such that the semi-trusted cloud service provider can decrypt the protected data to obtain the plaintext. Data confidentiality is not ensured as claimed. In this paper, we will explicitly indicate how this weakness damages Huang *et al.*'s scheme.

Keywords: Cloud Computing; Data Confidentiality; Data Collaboration; Hierarchical Attribute-based Encryption

1 Introduction

With rapid progress of network technologies, plenty of various applications and services are proposed and realized, and cloud computing revolutionizes the way how services are provided. Cloud computing possesses superior properties to benefit users such that resources including storage can be easily accessed, shared, and virtualized. Moreover, distributed computing is also allowed in cloud computing.

In addition to the above advantages, cloud computing can help users to save time and money because they do not need to construct the infrastructure by themselves completely. Cloud computing ensures flexibility. For example, users can obtain the required resources or services provided in the cloud and keep essential data secretly and locally. The flexible property makes more and more enterprises utilize cloud-based services.

Although cloud computing brings great benefits to enterprises and cloud users, many security issues are raised. Data confidentiality and access control in cloud computing are serious and urgent. It is because the cloud service provider (CSP) is semi-trusted and the data stored in the cloud may be disclosed by an unauthorized user or a malicious employee in CSP. This denotes that data leakage will take place if these security issues are not well and appropriately addressed [2]. As a result, data confidentiality and access control are important issues in cloud computing.

The reliable approach to protect data is encrypting data before being outsourced. The traditional methods for data encryption include symmetric encryption and asymmetric encryption. However, the above two traditional encryption methods are not suitable for data access control in cloud systems. As a result, attribute-based encryption (ABE) is proposed to ensure data access control with high precision [10]. An ABE mechanism enables access control over encrypted data with access policies and attributes among private keys and ciphertexts. Moreover, ciphertext-policy attribute-based encryption (CP-ABE) makes the data owner define access policies on all attributes that users need to decrypt the ciphertext. By CP-ABE, data confidentiality and data access control can be guaranteed [3].

However, the previous methods are designed to provide users with secure data reading while how multiple users collaboratively manipulate encrypted data in cloud computing is not taken into consideration.

Data collaboration service offered by CSP supports availability and consistency of the data shared among users [1]. In short, cloud computing is providing most of the functions originally provided by computers via the Internet. A user only needs one terminal to complete all functions such as the website setup, program development, and file storage. In order to realize and provide secure data collaboration services in cloud computing, only authorized users have the right to access or modify data in the cloud. That is, CSP needs to verify the user′s legitimacy. A cryptographic technique, Attribute-Based Signature (ABS), can help CSP verify the user when he/she requests to modify the data stored in the cloud. In an ABS system, the user can sign messages with his/her attributes key. Then, from the signature, CSP can check whether the signer′s attributes meet the access policy while the signer′s identity is unknown.

In recent years, many researches about the topics have been proposed. In 2011, Hur *et al.* proposed an attribute-based access control scheme in data outsourcing systems [5]. In 2012, Wan *et al.* proposed a hierarchical attribute-based access control in cloud computing scheme [8]. However, the above two schemes only take data sharing into consideration and cannot support write operations over stored data. In 2013, Li *et al.* proposed a secure sharing scheme based on attribute-based encryption for personal health records in cloud computing [6]. Li *et al.*′s scheme allows write operations. Unfortunately, the cloud still cannot verify the user′s write permission after receiving the re-encrypted modified data. In 2015, Yang *et al.* proposed one outsourcing scheme for big data access control in cloud and claimed that it cloud ensure security and verifiability [9]. Unfortunately, Liu *et al.* show that Yang *et al.*′s outsourcing scheme for big data access control in cloud suffers from some security flaws [7].

In 2017, Huang *et al.* proposed a data collaboration scheme with hierarchical attribute-based encryption in cloud computing [4]. Huang *et al.*′s scheme applies ABE, attribute-based signature (ABS), and bilinear map to ensuring data confidentiality and data access control. In their system model, there are five entities, central authority, domain authority, CSP, data owner, and user. The central authority, a trusted third party, manages domain authorities, sets up system parameters, and issues the secret parameter to the domain authority at the top level. A domain authority is a trusted third party, manages multiple domain authorities and domain users, and generates the master key for each domain authority at the next level and attribute secret keys for users. CSP, a semi-trusted party, provides data storage and collaboration service, offers partial decryption and partial signing, and is responsible for verifying the re-encrypted data before accepting it. The data owner outsources the encrypted data to CSP for collaboration. A user possessing a set of attributes satisfying the access policy can access and modify the data in cloud computing. Huang *et al.* also claimed that their scheme ensured data confidentiality. After analyzing Huang *et al.*′s scheme, we find that the semi-trusted cloud service provider can decrypt the protected data and obtain the plaintext after an authorized user modifies the data. That is, Huang *et al.*′s scheme cannot provide data confidentiality as claimed.

The rest of this paper is organized as follows. Section 2 reviews Huang *et al.*′s data collaboration scheme with hierarchical attribute-based encryption in cloud computing. Analyses on Huang *et al.*′s scheme are given in Section 3. At last, some conclusions are drawn in Section 4.

2 Review of Huang *et al.*′s Scheme

Huang *et al.*′s scheme is composed of six phases:

1) System setup phase;
2) Domain setup phase;
3) Key generation phase;
4) Data encryption phase;
5) Data decryption phase;
6) Data modification phase.

In this section, we first introduce the symbols and nine algorithms used in Huang *et al.*′s scheme. Then we review Huang *et al.*′s scheme. The details are as follows.

2.1 Notations

Notations used in Huang *et al.*′s scheme are listed in Table 1.

Table 1: Notations used in Huang *et al.*′s scheme

Symbol	Definition
CSP	Cloud service provider
PK	Central authority′s public key
MK	Entity′s master key
S	A set of attributes
SK	User′s attribute secret keys
AK	User′s attribute key
GK	Global key
T	Access policy
DK	Data encryption key
CT	Ciphertext
ST	Signature
Enc/Dec	Symmetric encryption/decryption

2.2 Algorithms

Huang *et al.* proposed nine algorithms and used them to define the designed system. The definitions of these nine algorithms are shown as follows.

1) $Setup(K)$. The central authority takes a security parameter K as input and outputs the system public key PK and the central authority′s master secret key MK_0.

2) *CreateDM*(*PK, MK_l, S*). The central authority or a domain authority takes *PK*, the master key MK_l and a set of attributes *S* as inputs and outputs the master secret key MK_{l+1} for the domain authority at the next level.

3) *KeyGen*(*PK, MK_l, S*). A domain authority takes *PK, MK_l* and *S* as inputs and outputs the attribute secret keys *SK* for each domain user.

4) *Encrypt*(*PK, M, T*). The data owner takes *PK*, a message *M* and an access policy *T* as inputs and outputs the ciphertext *CT*.

5) *PartDec*(*CT, AK*). A user uses *SK* to generate the attribute key *AK* and sends *AK* to *CSP*. *CSP* takes *CT* and *AK* as inputs. If the attributes in *AK* satisfy *T* in *CT*, *CSP* outputs a partial decrypted ciphertext CT_P.

6) *Decrypt*(CT_P, *SK*). A user takes CT_P and *SK* as inputs, recovers the data encryption key *DK*, and outputs the plaintext *M*.

7) *PartSign*(*Q, AK*). *CSP* takes a data collaboration request *Q* and *AK* as inputs and outputs a partial signature ST_P and a global key *GK*.

8) *Sign*(ST_P, *SK*). A user takes ST_P and *SK* as inputs and outputs the signature *ST*.

9) *Verify*(*T, ST, GK*). *CSP* takes *T, ST* and *GK* as inputs. If *ST* is the user′s valid signature such that *S* satisfies *T*, it outputs true.

2.3 System Setup Phase

In the beginning, the central authority executes *Setup* algorithm as follows:

Step 1. Selects a bilinear group G_1 of prime order p and generator g and the bilinear map $\hat{e} : G_1 \times G_1 \rightarrow G_2$.

Step 2. Selects random numbers α and β in Z_p and defines hash functions $H_1, H_2 : \{0, 1\}^* \rightarrow G_1$.

Step 3. Sets the master key $MK_0 = (\alpha, \beta)$ that is kept secret by the central authority and obtains the system public key PK, where $PK = (g^\alpha, g^\beta)$.

2.4 Domain Setup Phase

The central authority or a domain authority will be involved in this phase to execute $CreateDM$ algorithm. For clarity, two cases are given.

Case 1: The central authority executes $CreateDM$ algorithm as follows:

Step 1. Selects a unique number δ_l and chooses $\delta_{l,i} \in Z_p$ randomly for each attribute in A for $i = 1, 2, \ldots, m$, where A is a set of m attributes and $A = \{a1, a2, ..., a_m\}$.

Step 2. Computes $MK_l = (A, \overline{D_l} = g^{(\alpha+\delta_l)\beta}, \{\overline{D}_{l,i} = g^{\delta_l \beta} H_1(i)^{\delta_{l,i}}, \overline{D}'_{l,i} = g^{\delta_{l,i}} | a_i \in A, i \in \{1, 2, ..., m\}\})$ for the domain authority at the top level.

Case 2: The high level domain authority with MK_l executes $CreateDM$ algorithm as follows:

Step 1. Selects a unique number ε_l and chooses $\varepsilon_{l,i} \in Z_p$ randomly for each attribute in A' for $i = 1, 2, ..., n$, where A' is a set of n attributes $A' = \{a_1, a_2, ..., a_n\}$.

Step 2. Computes $MK_{l+1} = (A', \overline{D}_{l+1} = \overline{D}_l \cdot g^{\varepsilon_l \beta}, \{\overline{D}_{l+1,i} = \overline{D}_{l,i} \cdot g^{\varepsilon_l \beta} H_1(i)^{\varepsilon_{l,i}}, \overline{D}'_{l+1,i} = \overline{D}'_{l,i} \cdot g^{\varepsilon_{l,i}} | a_i \in A', i \in \{1, 2, ..., n\}\})$ for the domain authority at the next level.

2.5 Key Generation Phase

When a user joins in the domain, the corresponding domain authority with MK_l executes $KeyGen$ algorithm as follows:

Step 1. Selects $\gamma \in Z_p$ randomly for the user and chooses $\gamma_i \in Z_p$ randomly for each a_i in S, where S is a set of the user′s attributes.

Step 2. Computes the attribute secret keys $SK = (S, D = \overline{D}_l \cdot (g^\beta)^\gamma, \{D_i = \overline{D}_{l,i} \cdot g^{\gamma\beta} H_1(i)^{\gamma_i}, D'_i = \overline{D}'_{l,i} \cdot g^{\gamma_i} | i \in S\})$ for the user, where $i \in S$ is the shorthand for $a_i \in S$.

2.6 Data Encryption Phase

The data owner executes $Encrypt$ algorithm to encrypt the data M, defines the access policy T, and outsources the ciphertext to the cloud. The data owner performs as follows:

Step 1. Selects a random number $DK \in Z_p$ to encrypt the data M by using a symmetric encryption algorithm. Note that M will be encrypted under the access policy T.

Step 2. Generates a polynomial p_x for each node x in the access tree T with a top-down manner starting from the root node R. Sets the degree d_x of p_x to be $k_x - 1$ for each node x in T, where k_x is the threshold value of x and $k_x = 1$ if x a leaf node. On the root node R, chooses a random number $s \in Z_p$, sets $p_R(0) = s$, and chooses other d_R nodes randomly to define p_R. For other node x, sets $p_x(0) = p_{parent(x)}(index(x))$ and chooses other d_x nodes randomly to define p_x, where $index(x)$ is the label associated with x and $index(x)$ will be from 1 to $num(p)$ when x is the child node of node p and $num(p)$ denotes the number of p′s child nodes.

Step 3. Computes the ciphertext $CT = (T, E = Enc_{DK}(M), \tilde{C} = DK \cdot \hat{e}(g, g)^{\alpha\beta_s}, C = g^s, \{C_y = g^{p_y(0)}, C'_y = H_1(attr_y)^{p_y(0)}\}_{y \in Y})$ and outsources CT to CSP, where Y is a set of leaf nodes in access policy T.

2.7 Data Decryption Phase

This phase is composed of two parts, partial decryption phase and decryption phase. The details are as follows.

2.7.1 Partial Decryption Phase

When a user wants to access the data owner′s outsourced ciphertext from CSP, he/she first generates the attribute key $AK = \{D_i, D'_i | i \in S\}$ to CSP.

After getting AK, CSP executes $PartDec$ algorithm to partially decrypt the ciphertext. Then CSP executes a recursive algorithm, $DecryptNode$ algorithm. The recursive algorithm $DecryptNode(CT, AK, p)$ takes the ciphertext CT, the attribute key AK associated with S, and a node p from T as inputs.

If the node p is a leaf node y of T, $i = attr_y$, where $attr_y$ denotes an attribute associated with the leaf node y. If $i \in S$, $DecryptNode(CT, AK, p) = DecryptNode(CT, AK, y) = \frac{\hat{e}(D_i, C_y)}{\hat{e}(D'_i, C'_y)}$; otherwise, if $i \notin S$, $DecryptNode(CT, AK, y) = \perp$.

If the node p is a non-leaf node x of T, $DecryptNode(CT, AK, x)$ is executed by calling $DecryptNode(CT, AK, z)$ for all child nodes z of x and storing the output F_z. If no S_x, an arbitrary k_x-sized set of child nodes z of x such that $F_z \neq \perp$, exists, the node does not meet T and $DecryptNode(CT, AK, x) = \perp$. Otherwise, it denotes the subtree rooted at node x meets the access policy T if and only if k_x subtrees rooted at x′s children meet T. F_x is computed as follows, where $parent(z)$ is a parent node of z, $index(z)$ is the label associated with z, and $\Delta_{r,S_x}(x) = \prod_{j \in S_x, j \neq r} \frac{x-j}{r-j}$. Because z is a child node of x, $index(z)$ will be from 1 to $num(x)$, where $num(x)$ is the number of x′s children.

$$F_x = \prod_{Z \in S_x} F_Z^{\Delta_{j,S'_x(0)}} = \hat{e}(g, g)^{(\delta_l + \gamma)\beta p_x(0)}$$

where $S'_x = \{index(z) | z \in S_x\}$ and $j = index(z)$. With the recursive approach, calling $DecryptNode(CT, AK, R)$ can have the masking factor W efficiently obtained to decrypt CT such that $W = DecryptNode(CT, AK, R) = \hat{e}(g, g)^{(\delta_l + \gamma)\beta_s}$. Then, CSP sends the partial decrypted ciphertext $CT_P = (E, \tilde{C}, C, W)$ to the user.

2.7.2 Decryption Phase

After receiving CT_P, the user executes $Decrypt$ algorithm to retrieve the plaintext. The user first computes $DK = \tilde{C}/\hat{e}(C, D)/W)$ and obtains DK. Then the user can use DK to retrieve $M = Dec_{DK}(E)$.

2.8 Data Modification Phase

When a user needs to modify the stored data in the cloud to work collaboratively, he/she must use his/her attributes to sign the data collaboration request by using attribute-based signature, ABS. Only the user's signature satisfying the access policy can be authorized to outsource the re-encrypted data. Data modification phase is composed of four parts, writing data phase, partial signing phase, signing phase and verification phase. The details are as follows.

2.8.1 Writing Data Phase

The collaborative user obtains the plaintext in data decryption phase. After the user modifies the data, he/she re-encrypts data with T. Then the user sends the data collaboration request Q, AK and the re-encrypted data to CSP.

2.8.2 Partial Signing Phase

After receiving the collaboration request, CSP executes $PartSign$ algorithm. CSP selects a random number $\mu \in Z_p$ and computes $\tilde{S_0} = H_2(Q)^\mu$ and $S_0 = g^\mu$. CSP generates a polynomial q_x for each node x in the access tree T with a top-down manner starting from the root node R. CSP sets the degree b_x of q_x to be $k_x - 1$ for each node x in T, where k_x is the threshold value of x. On the root node R, CSP chooses a random number $t \in Z_p$, sets $q_R(0) = t$, and chooses other b_R nodes randomly to define q_R. For other node x, CSP sets $q_x(0) = q_{parent(x)}(index(x))$ and chooses other b_x nodes randomly to define q_x. CSP computes the global key $GK = \{K_y = g^{q_y(0)}, K'_y = H_1(attr_y)^{q_y(0)} | y \in Y\}$ for each $y \in Y$, where Y is a set of leaf nodes in access policy T.

CSP selects a random number $t_i \in Z_p$ for each $i \in Y$ and uses AK to compute $\{S_i, S'_i\}$, where $\{S_i = D_i H_1(i)^{t_i}, S'_i = D'_i g^{t_i} | i \in S \cap Y\}$ and $\{S_i = H_1(i)^{t_i}, S'_i = g^{t_i} | i \in Y/S \cap Y\}$. Then CSP generates the partial signature $ST_P = (\tilde{S_0}, S_0, \{S_i, S'_i | i \in Y\})$ and sends ST_P to the user.

2.8.3 Signing Phase

After receiving ST_P, the user executes $Sign$ algorithm to generate the signature. The user computes $\tilde{S} = \tilde{S_0} D$ and $S = S_0$ and generates the signature $ST = (\tilde{S}, S, \{S_i, Si' | i \in Y\})$. Then the user sends ST to CSP.

2.8.4 Verification Phase

After receiving ST, CSP executes $Verify$ algorithm to verify the signature ST. CSP executes $VerifyNode$ algorithm that is a recursive algorithm. The recursive algorithm $VerifyNode$ takes ST, a node p from T and GK associated with a set of attributes as inputs.

If the node p is a leaf node y of T, $i = attr_y$. If $i \in S \cap Y$, $VerifyNode(ST, GK, p) = VerifyNode(ST, GK, y) = \frac{\hat{e}(S_i, K_y)}{\hat{e}(S'_i, K'_y)}$. If $i \in Y/S \cap Y$, $VerifyNode(ST, GK, p) = VerifyNode(ST, GK, y) = \frac{\hat{e}(S_i, K_y)}{\hat{e}(S'_i, K'_y)} = 1$.

If the node p is a non-leaf node x, $VerifyNode(ST, GK, x)$ is executed by calling $VerifyNode(ST, GK, z)$ for all child nodes z of x

and storing the output G_z. If no G_z, an arbitrary k_x-sized set of child nodes z of x such that $G_z \neq \perp$, exists, the node does not meet T and $VerifyNode(ST, GK, x) = \perp$. Otherwise, it denotes the subtree rooted at node x meets the access policy T if and only if k_x subtrees rooted at x's children meet T. G_x is computed as follows, where $parent(z)$ is a parent node of z, $index(z)$ is the label associated with z, and $\Delta_{r,S_x}(x) = \prod_{j \in S_x, j \neq r} \frac{x-j}{r-j}$. Because z is a child node of x, $index(z)$ will be from 1 to $num(x)$, where $num(x)$ is the number of x's children.

$$G_x = \prod_{Z \in S_x} G_Z^{\Delta_{i,S_x'(0)}} = \hat{e}(g,g)^{(\delta_l+\gamma)\beta q_x(0)}$$

where $S_x' = \{index(z) | z \in S_x\}$ and $i = index(z)$. With the recursive approach, calling $VerifyNode(ST, GK, R)$ can have the masking factor I efficiently obtained to verify the signature such that $I = VerifyNode(ST, GK, R) = \hat{e}(g,g)^{(\delta_l+\gamma)\beta t}$. Then, CSP checks if $\frac{\hat{e}(g,\tilde{S})}{\hat{e}(H_2(Q),S)\cdot(I)^{1/t}}$ equals $\hat{e}(g,g)^{\alpha\beta}$. If they are equal, CSP accepts the signature and the re-encrypted data form the collaborative user. Otherwise, CSP rejects this data collaboration request.

3 Analysis on Huang *et al.*'s Scheme

After analyzing Huang *et al.*'s scheme, we find that their scheme cannot provide data confidentiality as claimed. Because the cloud service provider CSP is semi-trusted in Huang *et al.*'s scheme, CSP should neither know nor retrieve what the original data is even when users use data collaboration service. In Huang *et al.*'s scheme, the data M is protected by being encrypted by the data encryption key DK, and only users who meet the access policy can work collaboratively. Because it is only mentioned that the user modifies the data and re-encrypts data with T in writing data phase of data modification phase, this makes two cases possible. First, the modified data is re-encrypted with the same DK. Second, the modified data is re-encrypted with new DK by executing data encryption phase. No matter which case is true, CSP can obtain DK to retrieve data. For clarity, the details are given in the following.

3.1 Re-encrypting Data with The Same DK

Suppose the modified data is re-encrypted with the same DK in writing data phase. In partial decryption phase of data decryption phase, when a user wants to access the data owner's outsourced ciphertext from CSP, he/she first generates the attribute key $AK = \{D_i, D_i' | i \in S\}$ to CSP. After getting AK, CSP executes $PartDec$ algorithm to partially decrypt the ciphertext with a recursive algorithm, $DecryptNode$ algorithm. Then, CSP sends the partial decrypted ciphertext $CT_P = (E, \tilde{C}, C, W)$ to the user, where $W = DecryptNode(CT, AK, R) = \hat{e}(g,g)^{(\delta_l+\gamma)\beta_S}$. In decryption phase of data decryption phase, after receiving CT_P, the user executes $Decrypt$ algorithm to retrieve the plaintext by computing $DK = \tilde{C}/(\hat{e}(C,D)/W)$ and $M = Dec_D K(E)$. The above denotes that CSP is aware of $(E, \tilde{C}, C, W)$ after an authorized user accesses the data owner's outsourced ciphertext from CSP.

Suppose that U_1, who is an authorized user and has accessed the outsourced ciphertext from CSP, wants to modify the data. That is, data modification phase will be executed. In partial signing phase, CSP executes $PartSign$ algorithm by generating the partial signature $ST_P = (\tilde{S_0}, S_0, \{S_i, S_i' | i \in Y\})$ and sending ST_P to the user, where $\tilde{S_0} = H_2(Q)^\mu$ and $S_0 = g^\mu$. In signing phase of data modification phase, after receiving ST_P, the user executes $Sign$ algorithm to generate the signature by computing $\tilde{S_0} = \tilde{S_0}D$ and $S = S_0$ and generating the signature $ST = (\tilde{S}, S, \{S_i, S_i' | i \in Y\})$. In verification phase, after receiving ST, CSP executes $Verify$ algorithm to verify the signature ST. The above denotes that CSP is aware of $(\tilde{S_0}, S_0, \tilde{S}, S, \{S_i, S_i' | i \in Y\})$ after an authorized user wants to modify the data.

From then on, CSP knows $(E, \tilde{C}, C, W)$ and $(\tilde{S_0}, S_0, \tilde{S}, S, \{S_i, S_i' | i \in Y\})$. To retrieve the data, CSP performs as follows:

Step 1. Computes $\tilde{S} \times (\tilde{S_0})^{-1} = (\tilde{S_0}D) \times (\tilde{S_0})^{-1} = D$.

Step 2. Computes $DK = \tilde{C}/(\tilde{e}(C,D)/W)$.

Step 3. Computes $M = Dec_{DK}(E)$.

According to the above, it is obvious that CSP can retrieve the original data after an authorized user modifies the data. This found weakness shows that Huang *et al.*'s scheme cannot ensure data confidentiality.

3.2 Re-encrypting Data with New DK by Executing Data Encryption Phase

Suppose the modified data is re-encrypted with new DK by executing data encryption phase in writing data phase. If a user U_1 has ever modified the data, CSP can obtain D after signing phase is executed. When another authorized user U_2 wants to access the re-encrypted data, CSP can get DK with D to retrieve the data M. The details are as follows:

Step 1. In signing phase, U_1 receives $ST_P = (\tilde{S_0}, S_0, \{S_i, S_i' | i \in Y\})$ from CSP. Then U_1 computes $\tilde{S_0} = \tilde{S_0}D$ and $S = S_0$ and sends $ST = (\tilde{S}, S, \{S_i, S_i' | i \in Y\})$ to CSP. Because CSP is aware of $(\tilde{S_0}, S_0, \tilde{S}, S, \{S_i, S_i' | i \in Y\})$, CSP can retrieve D by computing $\tilde{S} \times (\tilde{S_0})^{-1} = D$.

Step 2. When U_2 accesses the re-encrypted data, U_2 receives $CT_P = (E, \tilde{C}, C, W)$ from CSP in decryption phase of data decryption phase. U_2 uses parameters $(\tilde{C}, C, W, D)$ to compute $DK = \tilde{C}/(\tilde{e}(C, D)/W)$. Then U_2 can retrieve the data M with DK. It means that CSP is also capable of computing the data encryption key DK because $\tilde{C}, C, W$, and D are all known. Thereupon, CSP can also retrieve the data M with DK.

According to the above, even if the modified data is re-encrypted with new DK by executing data encryption phase, CSP still can decrypt the encrypted data to retrieve the plaintext after another authorized user accesses the re-encrypted data.

4 Conclusions

Huang *et al.* proposed a hierarchical attribute-based encryption scheme to realize data collaboration in cloud computing. In this paper, we explicitly show how Huang *et al.*'s scheme suffers from one weakness. The data M is protected by the key DK, and it is supposed that only users who meet the access policy could obtain the plaintext. However, we find that CSP can retrieve the outsourced data after an authorized user modifies the data because CSP can get the data encryption key DK. Because CSP is semi-trusted, CSP should never know what the data M is. As a result, data confidentiality cannot be ensured in Huang *et al.*'s scheme. According to our findings, how to design a secure and efficient data collaboration scheme in cloud computing is still an urgent and tough issue.

Acknowledgments

This work was supported in part by Ministry of Science and Technology under the Grants MOST 106-2221-E-034-006-, MOST 106-2410-H-025-006-, MOST 106-2622-H-025-001-CC3, and MOST 107-2622-H-025-001-CC3.

References

[1] X. Dong, J. Yu, Y. Luo, Y. Chen, G. Xue and M. Li, "Achieving secure and efficient data collaboration in cloud computing," in *Proceedings of IEEE/ACM 21st International Symposium on Quality of Service (IWQoS'13)*, pp. 195–200, June 2013.

[2] Q. Huang, Z. Ma, Y. Yang, J. Fu and X. Niu, "Secure data sharing and retrieval using attribute-based encryption in cloud-based osns," *Chinese Journal of Electronics*, vol. 23, no. 3, pp. 557–563, 2014.

[3] Q. Huang, Z. Ma, Y. Yang, J. Fu and X. Niu, "Eabds: attribute-based secure data sharing with efficient revocation in cloud computing," *Chinese Journal of Electronics*, vol. 24, no. 4, pp. 862–868, 2015.

[4] Q. Huang, Y. Yang and M. Shen, "Secure and efficient data collaboration with hierarchical attribute-based encryption in cloud computing," *Future Generation Computer Systems*, vol. 72, pp. 239–249, 2017.

[5] J. Hur and D.K. Noh, "Attribute-based access control with efficient revocation in data outsourcing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 7, pp. 1214–1221, 2011.

[6] M. Li, S. Yu, Y. Zheng, K. Ren and W. Lou, "Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 1, pp. 131–143, 2013.

[7] L. Liu, Z. Cao and C. Mao, "A note on one outsourcing scheme for big data access control in cloud," *International Journal of Electronics and Information Engineering*, vol. 9, no. 1, pp. 29–35, 2018.

[8] Z. Wan, J. Liu and R. H. Deng, "Hasbe: a hierarchical attribute-based solution for flexible and scalable access control in cloud computing," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 2, pp. 743–754, 2012.

[9] K. Yang, X. H. Jia and K. Ren, "Secure and verifiable policy update outsourcing for big data access control in the cloud," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 12, pp. 3461–3470, 2015.

[10] S. Yu, C. Wang, K. Ren and W. Lou, "Achieving secure, scalable and fine-grained data access control in cloud computing," in *Proceedings of IEEE International Conference on Computer Communications (IEEE INFOCOM'10)*, pp. 1–9, Mar. 2010.

Biography

Wei-Liang Tai received the Ph.D. degree in computer science and information engineering from National Chung Cheng University, Taiwan, in 2008. He is currently Associate Professor, Department of Information Communications, Chinese Culture University. His main interests are in information security and forensics and multimedia signal processing. He is currently an Editor of KSII Transactions on Internet and Information Systems.

Ya-Fen Chang is a Professor of Department of Computer Science and Information Engineering at National Taichung University of Science and Technology in Taiwan. She received her BSc degree in computer science and information engineering from National Chiao Tung University and PhD degree in computer science and information engineering from National Chung Cheng University, Taiwan. Her current research interests include electronic commerce, information security, cryptography, mobile communications, image processing, and data hiding.

Wen-Hsin Huang received the MS degree in computer science and information engineering from National Taichung University of Science and Technology in Taiwan in 2018. Her main interests are in electronic commerce and information security.