SECURE MEDIA SHARING IN THE CLOUD: TWO-DIMENSIONAL-SCALABLE ACCESS CONTROL AND COMPREHENSIVE KEY MANAGEMENT

Changsha Ma and Chang Wen Chen

Dept. of Comp. Sci. and Eng., State Univ. of New York at Buffalo, Buffalo, NY, 14260, USA
changsha@buffalo.edu, chencw@buffalo.edu

ABSTRACT

Media sharing in cloud environment, which supports sharing media content at any time and from anywhere, is a promising paradigm of social interaction. However, it also brings forth security issues in terms of data confidentiality and access control on media data consumers with different access privileges. One promising solution is scalable media access control, which is capable of providing data confidentiality by encrypting the media data and issuing the key to only authorized data consumers. More importantly, it can empower the data distributor to provide the same media content with various quality levels to the consumers with different privileges. Traditional schemes without utilizing the cloud resources achieve scalable media access control by generating access keys using hash chains. Despite of their computational efficiency, such schemes suffer from various problems including vulnerability to user collusion attack in two-dimensional case, inflexible key distribution, and ambiguous key revocation strategy. In this paper, we propose a novel two-dimensional-scalable access control by generating access keys based on Attribute-Based Encryption (ABE) algorithm. Moreover, the proposed scheme can efficiently achieve comprehensive key management including key distribution and key revocation by fully exploiting the cloud. Security analysis shows that the proposed scheme is able to provide collusion resistance, as well as forward and backward secrecy. We have also evaluated the efficiency of the scheme through numerical analysis and initial implementation.

***Index Terms*—** ABE, two-dimensional scalable media, key management, access control

1. INTRODUCTION

Media sharing is a contemporary social interaction in which a data distributor generates a media content and share it with the data consumers. It has become increasingly popular with the prevalence of social networks and the increasing of network bandwidth. However, due to the highly dynamic nature in terms of user number, network bandwidth, and platforms, it is very difficult to allocate resources following the traditional client-server mode[1]. Shifting the application into the cloud environment will relief this constraint, since the cloud computing infrastructure provides abundant resources including processing, memory, and storage[2]. Furthermore, the omnipresence nature of the cloud supports the users to conveniently share media content at any time and from anywhere. This is particularly desirable for mobile users, who take up a large portion of the media sharing users. For these reasons, cloud computing has been considered an integral component in the media sharing application.

Just as many applications adopting cloud computing, this paradigm also suffers from security issues in terms of data confidentiality and access control. In this case, the data consumer may not want to share the media data with everyone, and may not trust the cloud either. Particularly, in media sharing application, there exists different relationships between data consumers and the data distributor. That is, data consumers may have different access privileges. In this case, scalable media access control is necessary to empower the data distributor to provide the media content with different quality levels, such as different resolutions and Signal Noise Ratio (SNR) levels, for the consumers with different privileges.

With the support of scalable media formats such as JPEG 2000 and H.264 scalable video coding (SVC), a media stream can be encoded into a base layer that provides the lowest quality, and enhancement layers for enhancing the quality of the media data[3]. Based on such data structure, a straightforward scheme for scalable media access control would be to encrypt each media layer by an individual access key, and then issue the specific access keys to the authorized users with the corresponding access privileges. However, this naive approach will cause huge cost for key distribution, since a set of access keys need to be sent for each authorized user.

In order to simplify the key distribution, related works in terms of scalable media access control usually divide one encryption key into several segments and generate access keys through one-way hash chains. Since lower level keys can be generated from higher level keys but the reverse is infeasible, only one or limited number of keys need to be sent to each user. Unfortunately, such solutions provide a chance for user collusion when the media units are organized following the two-dimensional scalable format. That is two users may collude with each other to generate a valid but illegal key for

This research is supported by NSF Grant 0915842.

unauthorized higher level access. Resisting such collusion attack will make the key management more complex. For example, in [4], the authors proposed to combine Diffie Hellman algorithm with the segmentation method to achieve the goal while sacrificing computational efficiency. In [5], the authors proposed to resist collusion attack by dividing the encryption key into more segments, resulting in tedious calculation for access key recovery. In [6, 7], the authors proposed to parse the access structure into chains based on partially order theory, which is relatively more efficient but also results in many key segmentations. More importantly, these schemes are not applicable for media sharing in the cloud, since they lack desired flexibility for the distributors to control who can access the media data by simply distributing access keys for each target consumer. Furthermore, it is still an unsolved issue for these schemes to revoke access privileges of data consumers.

A recent research [1] has reported a Multi-message Ciphertext Policy Attribute-Based Encryption (MCP-ABE) scheme to achieve fine-grained scalable media access control. Specifically, this scheme sets a scalable access policy for the scalable media data, and encrypts the access keys under the policy. Data consumers with proper attributes can obtain the specific access keys and then acquire the media data with the corresponding quality. This scheme is rather flexible in terms of key distribution, since the data distributor does not need to distribute access keys for each consumer. However, this scheme only considers the media data with one-dimensional scalability, and the access key generation still follows the hash chain method as mentioned. When extending the scheme into two-dimensional-scalable access control, collusion problem will occur again. Furthermore, it is not clear how the key revocation can be implemented under this scheme.

In this paper, we consider to solve several open issues as we have discussed by fully exploiting the cloud. We propose to perform access control for media sharing in the cloud by (1) generating the access keys in a scalable way based on Ciphertext Policy Attribute-Based Encryption (CP-ABE), instead of using the hash chains as in the related works; (2) making the cloud assist the data consumers to recover the corresponding access keys according to their attributes, instead of directly distributing the access keys to the data consumers; (3) enabling flexible revocation with the assist of the cloud. To sum up, the proposed scheme has all the desired features including two-dimensional-scalability with collusion resistance, efficient key distribution, and flexible key revocation.

The rest of the paper is organized as follows. In Section 2 we introduce some preliminaries related to the proposed scheme. In Section 3 we describe the proposed scheme in detail. Security and performance analysis are presented in Section 4. Finally, conclusions are summarized in Section 5.

2. PRELIMINARIES

2.1. Bilinear Map

Let G_0 and G_1 be two multiplicative cyclic groups of prime order p. Let g be a generator of G_0 and e be a bilinear map, $e : G_0 \times G_0 \to G_1$. Then e has the following properties:
Bilinearity: for all $u, v \in G_0$ and $a, b \in Z_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$
Non-degeneracy: $e(g, g) \neq 1$

2.2. CP-ABE

For CP-ABE scheme, a message is encrypted under an access policy and a secret key of a user is associated with a set of attributes. A user could decrypt the message only if his attributes satisfy the access policy. The CP-ABE scheme includes five algorithms that can be illustrated as follows:

Setup: The setup algorithm chooses a bilinear group G_0 of prime order p with generator g, and two random exponents $\alpha, \beta \in Z_p$. Public key PK and master key MK are returned as:$\{PK = G_0, g, h = g^{\beta}, f = g^{1/\beta}, e(g, g)^{\alpha}\}, \{MK = \beta, g^{\alpha}\}$.
Encryption: Given an input message M, this algorithm encrypts M under the access tree $\mathcal{T}$. Firstly, a polynomial p_x is chosen for each tree node x in a manner as: (1) Set the degree d_x of the polynomial p_x to be $d_x = k_x - 1$, where k_x is the threshold value of node x; (2) For the root node R, choose a random $s \in Z_p$ and set $p_R(0) = s$, and randomly choose other points of polynomial p_R; (3) For any other node x, set $p_x(0) = p_{parent(x)}(index(x))$, and randomly choose other points of p_x. Let L be the set of leaf nodes in $\mathcal{T}$, the ciphertext is given as

$$\begin{aligned} CT =& (\mathcal{T}, \tilde{C} = Me(g, g)^{\alpha s}, C = h^s, \\ & \forall i \in L : E_i = g^{p_i(0)}, E'_i = H(att(i))^{p_i(0)}) \end{aligned}$$

Key Generation: Taking a set of attributes S as input, the key generation algorithm outputs a secret key that identifies with the set. First, it selects a random $r \in Z_p$ and a random $r_i \in Z_p$ for every attribute in S. The key is computed as

$$\begin{aligned} SK =& (D = g^{(\alpha+r)/\beta}, \\ & \forall i \in S : D_i = g^r \cdot H(i)^{r_i}, D'_i = g^{r_i}) \end{aligned}$$

Delegation: A user could create a new secret key with a set $\tilde{S}$ ($\tilde{S} \subseteq S$) of attributes from his own, by randomly selecting $\tilde{r}$ and $\tilde{r}_k \forall k \in \tilde{S}$ and computing the new key as

$$\begin{aligned} \tilde{SK} =& (\tilde{D} = Df^{\tilde{r}}, \\ & \forall k \in \tilde{S} : \tilde{D}_k = D_k g^{\tilde{r}} H(k)^{\tilde{r}_k}, \tilde{D}'_k = D'_k g^{\tilde{r}_k}) \end{aligned}$$

Decryption: The decryption algorithm takes as input a message encrypted under an access policy, a secret key SK of a user, and the public key PK. It first calculates $e(g, g)^{rp_x(0)}$ for each leaf node x. Then it recursively computes the corresponding values for non-leaf nodes in a bottom-up manner using polynomial interpolation technique. If the attributes completely satisfy the access policy of the tree, it could compute the value for the root node as $A = DecryptNode(CT, SK, r) = e(g, g)^{rp_{R(0)}} = e(g, g)^{rs}$. Then the message could be obtained by computing $\tilde{C}/(e(C, D)/A) = \tilde{C}/(e(h^s, g^{(\alpha+r)/\beta})/e(g, g)^{rs}) = M$.

Please refer to [8] for more details of CP-ABE algorithm.

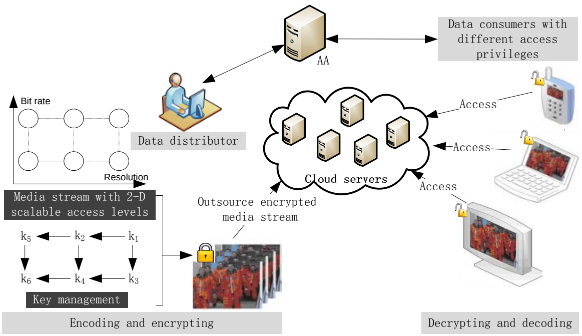


Fig. 1. System Architecture

3. THE PROPOSED SCHEME

3.1. System Architecture

As shown in Fig.1, the scalable media content sharing in cloud environment involves the following parties: the data distributor, the attribute authority (AA), the cloud servers, and the data consumers.

On the data distributor side, a media stream is encoded into a manner that is scalable in two dimensions. For example, Fig.1 shows a media stream that is encoded by SVC, and is scalable in terms of bit rate and resolution. The media data, including the base layer and enhancement layers, are then encrypted with specific access keys. Note that if a user could acquire the access key at a specific level, he can also acquire all of the access keys for every lower level. For instance, a user with key k_2 is capable to derive k_4, k_5, k_6 and decode all the media not exceeding the middle resolution level and the middle bit-rate level. The generation of the access keys will be discussed in details in section 3.2. The encrypted media stream is then output to the cloud server ready for sharing.

The AA is a trusted third party that sets up the system parameters of CP-ABE algorithm, and issues the secret key for each data consumer according to their attribute set.

Cloud servers handle the access requests from the data consumers, and assist in the process of key revocation. Note that cloud servers are not trusted by the cloud media users. Therefore, neither the access keys nor the media data could be obtained by the cloud servers without authorization.

A data consumer downloads the media of his interest from the cloud server, and decrypts the content by recovering the access keys with the assistance of the cloud server. Intuitively, we would like to enable the data consumers having a larger set of desired attributes to obtain the media data with a higher quality, and limit those who have a smaller set of attributes to access a lower quality media data. For example, a consumer with the attributes of "Tara's classmate, Tara's friend, same location with Tara" would be expected to have higher access privilege in terms of Tara's media data than the consumer only having the attribute "same location with Tara".

3.2. Scheme Description

Based on the structure of the encoded scalable media stream, we can build a scalable access tree according to the desired attributes for each media layer. Then we randomly select a key seed and encrypt it under the access tree using CP-ABE. Access keys can then be generated according to the key seed and the value for the corresponding tree nodes, which are denoted as key nodes. In the proposed scheme, we do not distribute any access keys directly to the users. Instead, we aim to use attributes to restrict the acquisition of the access keys. We enable a data consumer who is assigned with an attribute related secret key by the AA, to recover an access key when his attributes satisfy the corresponding part of the access tree. This process is assisted by the cloud server. However, none of the access keys could be obtained by the cloud. In the revocation case, the data distributor refreshes the parameters of the access tree, or additionally reselect some attributes, and relies on the cloud to perform the revocation. Specifically, the proposed scheme is composed of the following five parts.

3.2.1. System Setup

In this step, the AA runs the *Setup* algorithm of CP-ABE and outputs the public key PK and the master key MK.

3.2.2. Scalable Media Content Creation

A. Scalable Media Encoding

Given a media stream for sharing, the data distributor first encodes the media in a manner that is scalable in two dimensions. Based on the encoded media stream, we will have a desired access key management structure. For example, if the media stream is encoded into six layers, the corresponding key management structure will be like the one in Fig.2(a), which is composed of six access keys.

B. Access Key Generation

Let us treat the key management structure as a directed graph G. Then the edge $edge(k_i, k_j)$ indicates that the generation of k_j is based on k_i. It is also guaranteed that k_j can be obtained by a user if any $k_i(edge(k_i, k_j) \in G)$ could be obtained by the user. Unlike the traditional scalable media access control schemes, where k_j is the combination of the hashing components of k_i and is generated from k_i using hashing, here we generate k_j based on the desired attributes and the value for the key node that locates in the access tree and indicates k_i. Specifically, we adapt CP-ABE to build a scalable access policy for generating the access keys, in the following fashion.

(1) First, we convert the key management structure (a graph) into a binary tree, by keeping all nodes and edges in the graph and locating high level access keys at low layers of the access tree. Fig. 2 (a-b) show an example of such conversion;

(2) Next, we select a set of attributes for each of the edge, indicating a computable way from the higher level access key to the lower level access key when the attributes are satisfied, and construct a referencing tree, as shown in Fig. 2(c);

(3) Then, we construct a scalable access tree $\mathcal{T}$ based on the referencing tree. Nodes in $\mathcal{T}$ include leaf nodes, non-leaf nodes, and key nodes V_j that are related to access keys k_j.

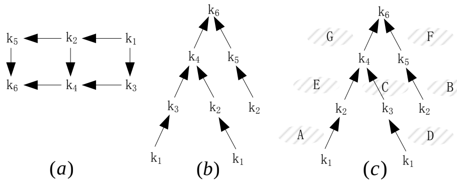


Fig. 2. The access key structure

From bottom to up, we iteratively construct subtrees for the key nodes, following the structure of the referencing tree. For instance, in Fig. 3 that is generated from the referencing tree shown in Fig. 2(c), we treat V_1 as a leaf node, use this and respectively choose attributes from attribute set A and D to construct a subtree for V_2 and V_3. We then iteratively construct the subtrees for V_4, V_5, V_6. Apart from the attribute sets in the referencing tree, another attribute $attri_{Common}$, which is the common attribute owned by all authorized consumers but not owned by the cloud server, is added as the child of the root node in $\mathcal{T}$. Then a polynomial p_x is chosen for each node x in $\mathcal{T}$ in the same way as CP-ABE. Since there may be two ways to recover an access key from the higher level access keys, some key node may appear twice in $\mathcal{T}$, either as a leaf node or as a non-leaf node. To keep the value of an access key consistent, we choose the same polynomial for the two repeated key nodes.

(4) Finally, we select a random value $m \in Z_p$ and generate the key seed $e(g,g)^m$, and then encrypt it under the access policy. Let L be the set of leaf node, the ciphertext will be

$$\begin{aligned} CT =& (\mathcal{T}, \tilde{C} = (e(g,g)^m)e(g,g)^{\alpha s}, C = h^s, \\ & \forall i \in L : E_i = g^{p_i(0)}, E'_i = H(att(i))^{p_i(0)}) \end{aligned}$$

After the construction of $\mathcal{T}$, we can generate the access keys by calculating $k_j = (K_j/e(g,g)^m) = e(g,g)^{r_{kj}}$, where $K_j = e(g,g)^{rp_{V_j}(0)}$ is the value for the key node V_j, and $r_{kj} = (r \cdot p_{V_j}(0) - m) \in Z_p$.

C. Media File Creation

After the access keys are generated, each layer is then encrypted by the corresponding access key. Standard encryption algorithms such as AES could be adopted. The encrypted media layers, the syntax of the access tree, and the values for the key nodes, i.e. K_j, are then sent by the data distributor to the cloud server. The cloud server will store these information and control the access on the encrypted media data.

3.2.3. User Registration

For a data consumer in the media sharing networks, the AA assigns him with a unique identity id, and issues a secret key according to his attributes for him. The secret key is defined as follows.

$$\begin{aligned} SK = \{ & D = g^{(\alpha+r)/\beta}, \\ SK_{Common} : & D_c = g^r \cdot H(c)^{r_c}, D'_c = g^{r_c}, \\ SK_{attri} : & (\forall i \in S : D_i = g^r \cdot H(i)^{r_i}, D'_i = g^{r_i}) \} \end{aligned}$$

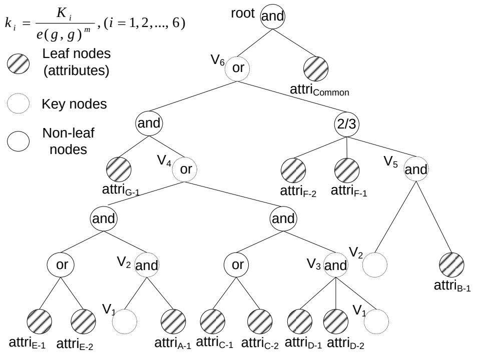


Fig. 3. Scalable access tree

Furthermore, the AA signs on the tuple (id, S) and sends $id, \sigma(id, S)$ with the secret key to the data consumer.

3.2.4. Access Control

A. Key distribution

When accessing a media stream stored in the cloud server, the data consumer sends $id, S, \sigma(id, S)$ along with the access request to the cloud server. On receiving the request, the cloud will first check the validity of the signature, and check whether the identity is revoked by the AA. If not, the cloud server will further check the attributes of the consumer and figure out the corresponding K_i needed to be sent to the consumer. For example, in Fig.3, if the attributes of the consumer include $attri_{G-1}$, $attri_{F-1}$, $attri_{F-2}$, $attri_{Common}$, the cloud will send K_4 and K_5 to the consumer. For the consumer whose attributes include $attri_{E-1}$, $attri_{E-2}$, $attri_{G-1}$, $attri_{Common}$, the cloud would send K_2 to him.

B. Key recovery

In CP-ABE, a user decrypts the message encrypted under the access tree by firstly computing the values for each leaf node, and then computing the values for the non-leaf nodes in a bottom-up manner. In the proposed scheme, however, not each authorized consumer has all of the satisfied attributes and hence could not compute the values of each leaf node. To decrypt the key seed, the data consumer needs the K_i sent by the cloud server. Then he carries out the following operations:

(1) Use SK_{attri}, SK_{Common} to recover the value F_x for each leaf node x corresponding to $attri_i \in S$:

$$\begin{aligned} F_x &= \frac{e(D_{attri_i}, E_{attri_i})}{e(D'_{attri_i}, E'_{attri_i})} \\ &= \frac{e(g^r \cdot H(attri_i)^{r_{attri_i}}, g^{p_x(0)})}{e(g^{r_{attri_i}}, H(attri_i)^{p_x(0)})} \\ &= \frac{e(g^r, g^{p_x(0)}) \cdot e(H(attri_i)^{r_{attri_i}}, g^{p_x(0)})}{e(g^{r_{attri_i}}, H(attri_i)^{p_x(0)})} \\ &= e(g,g)^{rp_x(0)} \end{aligned}$$

(2) Start from the tree level that K_i locates in, the data consumer computes the value F_y for each non-leaf node y, and

the value $K_j(j \geq i)$ for each key node V_j in bottom-up manner, as follows:

$$\begin{aligned} F_y &= \prod_{z \in S_y} F_z^{\triangle_{j,S'_y}(0)} (S_y\text{: the set of children node } z \text{ of node } y) \\ &= \prod_{z \in S_y} \left(e(g,g)^{rp_z(0)}\right)^{\triangle_{j,S'_y}(0)} \\ &= \prod_{z \in S_y} \left(e(g,g)^{rp_y(index(z))}\right)^{\triangle_{j,S'_y}(0)} \\ &= e(g,g)^{rp_y(0)} (S'_y = index(z) : z \in S_y) \end{aligned}$$

The calculation of K_j follows the same way as that of F_y.

(3) The consumer carries on the operation in step (2) until he obtains $F_{root} = e(g,g)^{rs}$. The result will be used to recover the key seed, since $\tilde{C}/(e(C,D)/F_{root}) = \tilde{C}/(e(h^s, g^{(\alpha+r)/\beta})/e(g,g)^{rs}) = e(g,g)^m$.

(4) The access keys can be computed according to $k_j = (K_j/e(g,g)^m)$. The corresponding media layers can also be decrypted.

3.2.5. Key Revocation

Access key revocation is needed when the data distributor would like to take away the access privilege of some data consumers in terms of the new media contents. There are two cases of key revocation, namely, revoking the access privilege of one specific data consumer and revoking specific attributes. In the first case, the data distributor needs to (1) reselect the polynomials for the nodes in $\mathcal{T}$, and refresh E_i as $g^{p'_i(0)}$ and E'_i as $H(att(i))^{p'_i(0)}$ in the ciphertext; (2) recompute access keys following $k'_i = K'_i/e(g,g)^m = e(g,g)^{r \cdot p'_{V_i}(0)}/e(g,g)^m$, and re-encrypt the media layers; (3) transmit the revoked id, and the refreshed ciphertext and K'_i to the cloud server. In the second case, the data distributor will additionally refresh the access policy by reselecting new attributes and replacing the revoked ones. In this case, E'_i will be refreshed as $H(att(i)')^{p'_i(0)}$.

Upon receiving the revocation message from the data distributor, the cloud server will update the ciphertext and K_i, and add the revoked id into the revocation list (RL). When a revoked user whose id is included in RL sends a new request to the cloud server, the cloud server will not respond him with the corresponding K_i, even if the user's attributes still satisfy the access policy. For the revoked users whose attributes no longer satisfy the access policy, the cloud server can easily identify it according to the syntax of the new $\mathcal{T}$, and then refuse the request.

4. ANALYSIS ON THE PROPOSED SCHEME

4.1. Security Analysis

4.1.1. Fine-grainedness of Access Control

The proposed scheme allows a data distributor to define fine-grained access policy for the media data by integrating attributes with access key structure. This is very much desired for sharing of social media and cloud media in which multiple levels of access privileges are needed.

4.1.2. Data Confidentiality

The media data for sharing is encrypted under symmetric encryption algorithm. CP-ABE is applied to compute the key of the symmetric encryption algorithm. The confidentiality is achieved based on the security of symmetric encryption algorithm and the security of CP-ABE. Besides, although the cloud server have all of the K_i, it cannot decrypt the key seed if it does not own the common attribute. Therefore, the cloud server shall not be able to recover the access keys, and hence could not obtain the media content, either.

4.1.3. Forward Secrecy

By changing the polynomials for the tree nodes, K_i can be changed, and hence the access keys can also be changed. In order to recover the new access keys to decrypt the new media data, the revoked consumer will need to acquire K'_i, which is equal to $e(g,g)^{r \cdot p'_{V_j}(0)}$. However, the data consumer only has $K_j = e(g,g)^{r \cdot p_{V_j}(0)}$. Without knowing the value of $p'_{V_j}(0)$ and $p_{V_j}(0)$, which are randomly chosen and kept private by the data distributor, there is no way for the consumer to compute K'_j from K_j. Therefore, we can claim that the revoked data consumer cannot acquire the subsequent access keys, given that he has all old access keys. As a result, forward secrecy is satisfied in the proposed scheme.

4.1.4. Backward Secrecy

When a new data consumer joins in the system, he will be assigned with a secret key related to his attributes. If the attributes satisfy part of the scalable access policy, he can obtain the current K'_j and the key seed, then recover k'_j. In order to recover the previous old access keys, the consumer has to compute K_j according to K'_j, which is infeasible according to the discussion in forward secrecy. This means that the proposed scheme also satisfies backward secrecy.

4.1.5. Collusion Resistance

For two authorized users each can only obtain access key at a specific level, it is impossible for them to collude with each other to recover a higher level access key, due to the lack of specific attributes. For example, consider the two consumers who are assigned K_2 and K_3, respectively, they cannot collude to acquire K_1, since neither of them has the required attributes $attri_{A-1}, attri_{D-1}, attri_{D-2}$.

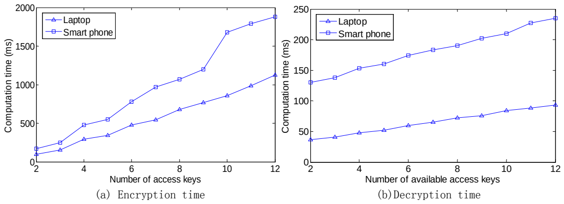
4.2. Performance Analysis

In this section, we present the evaluation on the effectiveness of the proposed scheme in terms of computation cost and communication cost. Specifically, we consider only the computation cost on the user side, but not the cost on the cloud side, since we assume the cloud has plentiful power.

4.2.1. Computation Cost

A. Computation Cost for Encryption

To encrypt the scalable media data, a data consumer needs to 1) run the encryption algorithm of CP-ABE; 2) generate access keys from K_j; 3) encrypt the media data using the access keys. As we indicated early, we shall not discuss the third part in detail. Therefore, we only evaluate the computation cost of the first two parts. The cost is proportional



(a) Encryption time

(b)Decryption time

Fig. 4. Computation cost on media sharing users

to the number of access keys. By assigning one to three attributes for each attribute set, we shall implement the encryption algorithm of CP-ABE as well as the multiplication on G_1, on a laptop HP520 (Intel Core Duo 1.6GHz) and a smart phone SAMSUNG (Dual-core 1.2GHz). The relationship between the computation time and the number of access keys can therefore be obtained. The results are shown in Fig.4(a).

B. Computation Cost for Decryption

The decryption operations on the data consumer side include 1) decrypting the key seed, 2) recovering the access keys, 3) and decrypting the media data. Since the media data decryption is not related to the proposed scheme, we only consider the first two steps. Specifically, key seed decryption is based on the adjusted decryption algorithm of CP-ABE, and the access key recovery needs multiplication operations on G_1. The running time of the two operations are both proportional to the number of access keys the data consumer could acquire. Furthermore, the running time of key seed decryption also depends on the structure of the access tree. By assigning one to three attributes for each attribute sets, we can obtain the average decryption time as shown in Fig.4(b).

C. Computation Cost for Revocation

Since the key seed remains the same in the revocation case, the data distributor does not need to re-encrypt it under the access tree. The computational operations will be N times of multiplications on G_1 and symmetric encryptions, where N denotes the number of system access keys. Note that multiplication on G_1 performs very efficiently (about 1ms) on the implement devices due to the lightweight computation.

Although the proposed scheme has some loss in efficiency comparing to related schemes that use only lightweight hashing, it can provide comprehensive key management and collusion-resilience that the related works are unable to offer.

4.2.2. *Communication Cost*

A. Communication Cost for Access Control

We consider the communication cost in terms of sending request to the cloud server and responding to the access request of the data consumers. Firstly, the request message sent by the consumer to the cloud include $id, S, \sigma(id, S)$. Both id and $\sigma(id, S)$ have the fixed size, while the size of attribute set S varies with the number of attributes owned by the consumer and the length of each attribute. The message size of the response sent by the cloud server is at most twice of the size of K_j, which is 48 bytes in the proposed scheme. The communication cost C is determined by both the message size and the bandwidth B for a specific entity.

B. Communication Cost for Revocation

The revocation message will include either id, $(E_i, E'_i)_{\forall i \in L}$, $S = \{K_j\}_{|S|=N}$ or $(E_i, E'_i)_{\forall i \in L}, S = \{K_j\}_{|S|=N}$. Therefore, the message size is about $(144N)$ bytes.

5. CONCLUSION

We have presented in this paper a novel scheme that exploits the ability of the cloud to achieve two-dimensional-scalable access control and comprehensive key management for media sharing in cloud environment. With this scheme, we are able to achieve several desirable security features including fine-grained access control, data confidentiality, forward and backward secrecy, and collusion resistance. We have confirmed the effectiveness of the proposed scheme through both numerical analysis and mobile terminal implementation with typical laptop and smart phones.

6. REFERENCES

[1] Wu, Y.; Zhuo, W.; Deng, R., "Attribute-Based Access to Scalable Media in Cloud-Assisted Content Sharing Networks," Multimedia, IEEE Transactions on , vol.15, no.4, pp.778,788, June 2013.

[2] Zhu, W.; Luo, C.; Wang, J. ; Li, S., "Multimedia Cloud Computing," Signal Processing Magazine, IEEE, vol.28, no.3, pp.59,69, May 2011.

[3] Schwarz, H.; Marpe, D.; Wiegand, T., "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard," Circuits and Systems for Video Technology, IEEE Trans. on, vol.17, no.9, pp.1103,1120, Sept. 2007.

[4] Zhu, B.B.; Feng, M.; Li, S., "An efficient key scheme for layered access control of MPEG-4 FGS video," ICME '04. 2004 IEEE International Conference on , vol.1, no., pp.443,446 Vol.1, 30-30 June 2004.

[5] Imaizumi, S.; Fujiyoshi, M.; Abe, Y.; Kiya, H., "Collusion Attack-Resilient Hierarchical Encryption of JPEG 2000 Codestreams with Scalable Access Control," Image Processing, 2007. ICIP 2007. IEEE International Conference on , vol.2, no., pp.II - 137,II - 140, Sept. 16 2007-Oct. 19 2007.

[6] Zhu, X.; Chen, C. W., "A collusion resilient key management scheme for multi-dimensional scalable media access control," Image Processing (ICIP), 2011 18th IEEE International Conference on , vol., no., pp.2769,2772, 11-14 Sept. 2011.

[7] Crampton, J.; Daud, R.; Martin, K. M., "Constructing Key Allignment Schemes from Chain Partitions", Proceedings of the 24th annual IFIP WG 11.3 working conference on Data and applications security and privacy, 2010.

[8] Bethencourt, J.; Sahai, A.; Waters, B., "Ciphertext-Policy Attribute-Based Encryption," Security and Privacy, 2007. SP '07. IEEE Symposium on , vol., no., pp.321,334, 20-23 May 2007.