This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2707126, IEEE Access

A Collaborative Key Management Protocol in Ciphertext Policy Attribute-Based Encryption for Cloud Data Sharing

Guofeng Lin, Hanshu Hong, and Zhixin Sun

***Abstract*— Ciphertext policy attribute-based encryption (CP-ABE) is a promising cryptographic technique for fine-grained access control of outsourced data in the cloud. However, some drawbacks of key management hinder the popularity of its application. One drawback in urgent need of solution is the key escrow problem. We indicate that front-end devices of clients like smart phones generally have limited privacy protection, so if private keys are entirely held by them, clients risk key exposure that is hardly noticed but inherently existed in previous research. Furthermore, enormous client decryption overhead limits the practical use of ABE. In this work, we propose a collaborative key management protocol in CP-ABE (CKM-CP-ABE). Our construction realizes distributed generation, issue and storage of private keys without adding any extra infrastructure. A fine-grained and immediate attribute revocation is provided for key update. The proposed collaborative mechanism effectively solves not only key escrow problem but also key exposure. Meanwhile, it helps markedly reduce client decryption overhead. A comparison with other representative CP-ABE schemes demonstrates that our scheme has somewhat better performance in terms of cloud-based outsourced data sharing on mobile devices. Finally, we provide proof of security for the proposed protocol.**

***Index Terms*—Cloud data sharing, CP-ABE, Key management, Security, fficiency.**

I. Introduction

With cost-effectiveness improvements in computational technology and large-scale networks, sharing data with others becomes correspondingly more convenient. Additionally, digital resources are more easily obtained via cloud computing and storage. Since cloud data sharing requires off-premises infrastructure that some organizations jointly held, remote storage are somehow threatening privacy of data owners. Therefore, enforcing the protection of personal, confidential and sensitive data stored in the cloud is extremely crucial [23][25][26]. The simultaneous participation of a large number of users requires fine-grained access control for data sharing. Attribute-based encryption (ABE) is a promising cryptographic primitive [21] that offers an interesting solution to secure and flexible data sharing. ABE has an inherent one-to-many property, which means a single key can decrypt different ciphertexts or different keys can decrypt the same ciphertext. There are two types of ABE, called ciphertext policy ABE (CP-ABE) and key policy ABE (KP-ABE). For CP-ABE, the access policy is embedded into a ciphertext and the attribute set is embedded into a private key. For KP-ABE, the access policy is embedded into a private key and the attribute set is embedded into a ciphertext. CP-ABE [2] allows data owners to define their own access policy. Anyone who wants to obtain data must first match the access policy attribute set. Due to this property, CP-ABE is quite suitable for the construction of secure, fine-grained access control for cloud data sharing [27][28].

However, there are still a lot of open challenges concerning the practical realization of ABEs especially in terms of private key management. For large numbers of previous ABE schemes [2][3][4][5][6][7], the key authority must be completely trustworthy, as it can decrypt all the ciphertext using a generated private key without permission of its owner. This is commonly called the key escrow problem and is an inherent disadvantage that threatens user privacy. Together with the growth of mobile applications, mobile cloud services [24] [31] have been introduced as a potential trend in cloud computing. Current research work hardly notices that mobile front-end devices, such as smartphones, are far more vulnerable than servers with respect to privacy protection [20]. Thus, the vulnerability in private key protection may easily lead to the exposure of keys to unauthorized users [30]. In addition, current ABE key management schemes also require much bilinear pairing calculation, exponentiation and multiplication, especially in the decryption step [5][29]. The resulting run time may be horribly unacceptable.

In this paper, we propose a novel collaborative key management protocol in ciphertext policy attribute-based encryption (CKM-CP-ABE) aiming to enhance security and efficiency of key management in cloud data sharing system. The main contributions are summarized as follows:

1) A novel collaborative protocol is presented. With help of

This work was supported in part by the Key Lab of Broadband Wireless Communication and Sensor Network Technology, Nanjing University of Posts and Telecommunications. The work of G. Lin was supported in part by National Natural Science Foundation of China (61672299, 61373135). The work of H. Hong was supported in part by National Natural Science Foundation of China (61170276). The work of Z. Sun was supported in part by National Natural Science Foundation of China (60973140).

G. Lin, H. Hong, and Z. Sun are with the Key Lab of Broadband Wireless Communication and Sensor Network Technology, Nanjing University of Posts and Telecommunications, Nanjing 210003, China (e-mail: 2015070249@njupt.edu.cn; 2014070244@njupt.edu.cn; sunzx@njupt.edu.cn)

interaction among the key authority, a cloud server and a client who tends to access data, distributed generation, issue and storage of private keys are realized. Thus, secure key management is guaranteed without adding any extra physical infrastructure, which is more easy to deploy compared with previous multi-authority schemes.

2) We introduce attribute groups to build the private key update algorithm. A unique attribute group key is allocated to each attribute group that contains clients who share the same attribute. Via updating attribute group key, a fine-grained and immediate attribute revocation is provided.
3) We indicate that not only key escrow problem but also key exposure is threatening the confidentiality of private keys, which is hardly noticed in previous research. Compared to previous key management protocols for attribute-based data sharing system in cloud, our proposed protocol effectively addresses both two problems by its collaborative key management. Finally, we provide proof of security for the proposed protocol.
4) The collaborative mechanism helps markedly reduce client decryption overhead by employing a decryption server to execute most of decryption while leave no knowledge about information to it.

The rest of paper is organized as follow: We review previous work on ABE primitives and their applications in Section II, give essential preliminaries in Section III, present the CKM-CP-ABE model for cloud data sharing in Section IV, propose main key management algorithm in Section V, provide comprehensive analysis on the security and efficiency of CKM-CP-ABE in Section VI, and draw conclusion and highlight our future work in Section VII.

II. Related Work

In 2005, Sahai and Waters [1] proposed a fuzzy identity-based encryption (FIBE) based on classic identity-based encryption. The identity of a receiver is represented by a set of attributes, which is embedded into his/her private key. If and only if the distance between the attribute set of receiver and the one of sender is shorter than a threshold, a receiver can extract the plaintext correctly. Since FIBE indicated some many key features of ABE, it laid a theoretical foundation of subsequent research into ABE. In 2006, Goyal *et al.* [21] proposed a formal definition of ABE. By introducing access tree, they built a fine-grained access policy for ABE. The research work indicated that their construction based on FIBE is a key policy ABE (KP-ABE), which means each private key is associated with an access policy and each ciphertext is associated with a set of attributes. There must be another type of ABE called ciphertext policy ABE (CP-ABE). For CP-ABE, each ciphertext is associated with an access policy and each private key is associated with a set of attributes. Inspired by reference [21], Bethencourt *et al.* [2] proposed a concrete construction of CP-ABE, in which a data sender can flexibly define the access policy before data is encrypted. Consequently, their CP-ABE guarantees not only data confidentiality but also realization of autonomic access control. The research demonstrated that CP-ABE is more suitable for construction of data outsourced system than KP-ABE. Cheung and Newport [13] proposed a novel access tree contain only AND gates with positive and negative attributes to enhance expressiveness of ABE access policy. At mean time, they introduced one-time signature to prove that their CP-ABE is secure under chosen ciphertext attacks (CPA-secure) with the reduction to Decisional Bilinear Diffie-Hellman Assumption (DBDH).

Attrapadung *et al.* [3] investigated ABEs' revocation and proposed a hybrid-revocable attribute-based encryption (HR-ABE) with the combination of direct and indirect revocation. When implementing encryption, each data sander is allowed to select which revocable scheme is used that combine advantages of both methods. Noting that its hybrid revocation has no effect on decryption although each data receiver has only one private key. Based on Bethencourt *et al.*'s CP-ABE [2], Hur *et al.* [14] proposed a novel attribute-based access control with fine-grained key update mechanism by introducing attribute group key. Their efficient scheme supports more flexible attribute revocation and user revocation, which enhanced forward and backward secrecy. Chandar *et al.* [11] concentrate their work on efficiency of revocation and proposed a hierarchical ABE (HABE). In their construction, lazy re-encryption was introduced so that only message to be updated will be re-encrypted. The analysis demonstrated HABE markedly reduces global computation overhead for attribute revocation and user revocation.

Waters [4] proposed a high efficient CP-ABE that first makes ciphertext size, encryption time and decryption time increase linearly with the complexity of access policy. In addition, the proposed scheme is proven to be selectively secure with the reduction to Decisional Parallel Bilinear Diffie-Hellman Exponent Assumption (DPBDHE). Green *et al.* [5] indicated that ciphertext size and decryption cost are major drawbacks of practical ABE applications. To overcome these problems, they proposed a novel ABE with outsourced decryption (OD-ABE) for both CP-ABE and KP-ABE that sets a proxy server for executing most of decrypting computation. When implementing decryption, a data receiver transfers a transform key and the ciphertext to a proxy server and receives a ElGamal style ciphertext. Subsequently, the plaintext can be extracted via very simple computation by the data receiver. With the potential trend of mobile cloud service, applying outsourced decryption scheme markedly optimizes user experience. Lai *et al.* [6] proposed an attribute-based encryption with verifiable outsourced decryption (VOD-ABE). In OD-ABE, the untrusted proxy server risks the confidentiality of ciphertext and transform keys. Furthermore, wrong computation may result in unavailability of the whole system. Their research work demonstrated the necessity of a proposed security requirement for OD-ABE called outsourcing verifiability. Lin et al. [7] proposed an improved VOD-ABE based on key encapsulation mechanism (KEM). Their experiment demonstrated that ciphertext size, cost of encryption and decryption are nearly half of those in Lai et al.'s constructions [6].

Chase *et al.* [8] proposed a novel key generation algorithm based on multi-authority ABE. They assume that central authority in previous multi-authority ABEs is semi-trusted so that can extract plaintext without permission of other authorities and users. The proposed algorithm requires interactions among central authority and other authorities to jointly generate keys with their master secrets so that the key escrow problem can be addressed. Pletea *et al.* [32] introduced a concept of hierarchical generalized attributes based on the global attribute collection, and proposed a hierarchical multi-authority mechanism for CP-ABE. When a user defines a access structure and requests data encryption, each key generation center (KGA) generates corresponding access policy and private key of its level so that security of key management is guaranteed even there exist a compromised KGA engaged in key issuing. Xu *et al.* [33] proposed a proxy re-encryption scheme for multi-authority ABE to realize efficient and fine-grained revocation. For key management in their scheme, access policies defined by users are mapped to a weighted access list, which reduces computation overhead during generating and issuing private keys. Zhang *et al.* [9] first introduced a third infrastructure in single-authority KP-ABE to solve key escrow problem. Its interaction overhead for key generation is far lighter than previous escrow-free multi-authority schemes. Hur [10] proposed a secure and efficient attribute-based data sharing system. Without adding any infrastructure, they provided a novel solution to key escrow problem in single-authority system by introducing two-party computation (2PC) between key generation center and data storing center. Easwormoorthy *et al.* [34] presented a key management infrastructure (KMI) for personal health records (PHR) cloud. They defined a private domain and a public domain to respectively share PHR with different users. In private domain, multi-authority CP-ABE is introduced for efficient access control of large number of users based on their profession roles. In public domain, decentralized KP-ABE for delegating access right to several users that are personally associated with PHR owners. Their proposed KMI aims to realize highly efficient attribute revocation and user revocation. Oualha *et al.* [35] indicated that despite tremendous computation resources are needed in ABE, lots of heavy computation can be done in advance. Considering limitation of energy and computation of nodes in Internet of Things, they proposed a novel CP-ABE by introducing a pre-computation technique, which computes and stores some essential elements before encryption occurs. Although real-time computation overhead is markedly decreased, their scheme requires trusted entities to store elements. Furthermore, trusted channels are also required to securely transfer those elements to desired nodes.

As mentioned above, previous schemes of key management in attribute-based data sharing system mainly focuses on key update, proxy re-encryption and outsourced decryption. Some research demonstrated untrusted key authority may lead to key escrow problem and provided corresponding solutions. However, little research notices that if authority is untrusted, front-end devices especially mobile ones must be far more untrusted than it because they are inherently vulnerable to illegal access. If private keys are still entirely stored in front-end devices, a worse problem called key exposure occurs threatening confidentiality of private keys. In addition, most of attribute-based data sharing schemes enhanced security of key management at the cost of decryption overhead of data receivers. Therefore, we are not satisfied with previous schemes of key management in terms of either security or efficiency.

III. PRELIMINARIES

A. Bilinear Pairings

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two multiplicative cyclic groups with prime order p, and let g be a generator. The map $e:\mathbb{G}_1\times\mathbb{G}_1\to\mathbb{G}_2$ is a bilinear map if the following two properties hold:

1) **Bilinearity:** For arbitrary $u,v\in\mathbb{G}_1$ and $a,b\in Z_p^*$, we have $e(u^a,v^b)=e(u,v)^{ab}$.
2) **Non-degeneracy:** There exists a generator g of $\mathbb{G}_1$ such that $e(g,g)\neq 1$ holds.

Note that e is a symmetric map, that is, $e(u^x,v^y)=e(u,v)^{xy}=e(u^y,v^x)$.

B. Linear Secret Sharing Scheme

Linear secret sharing scheme (LSSS) is an effective tool for building an access structure. We adopt the formal definition from [3] below:

Definition 1 (Access Structure): Let $P=\{p_1,p_2,\cdots,p_m\}$ be a set of participants. A collection $\mathbb{A}\subseteq 2^P$ is monotone if for arbitrary B and C we have that $B\in\mathbb{A}$ and $B\subseteq C$, and $C\in\mathbb{A}$ holds. An access structure (monotone access structure) is a collection (monotone collection) $\mathbb{A}\in 2^P\setminus\{\varnothing\}$. We call sets in $\mathbb{A}$ authorized sets, and sets not in $\mathbb{A}$ unauthorized sets.

Definition 2 (Linear Secret sharing scheme): Let P be a set of participants and M be a matrix with m rows and d columns. The map $\rho:\{1,2,\cdots,m\}\to P$ associates each row with one participant for labeling. A secret sharing scheme Π over P for access structure $\mathbb{A}$ is a linear secret sharing scheme in Z_p^* represented by (M,ρ) if it consists of two polynomial-time algorithms:

1) $Share(M,\rho)$: The share algorithm takes $s\in Z_p^*$ as an input secret to be shared. *Share* randomly chooses a group of elements $y_2,y_3,\cdots,y_d\in Z_p^*$ and generates a column vector $\boldsymbol{v}=(s,y_2,y_3,\cdots,y_d)$. Then, it outputs $M\cdot\boldsymbol{v}$ as a vector that l participants share such that they each possess an element. We define M_i as the ith row in M so that $\lambda_{\rho(i)}=M_i\cdot\boldsymbol{v}$ is the element belonging to participant $\rho(i)$.
2) $Recovery(S)$: The recovery algorithm takes a set S of participants as input. We define a set $I=\{i\mid\rho(i)\in S\}$. If $S\in\mathbb{A}$, there exists a group of constants $\{\omega_i\mid i\in I\}$, which

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2707126, IEEE Access

has a linear property:

$$\sum_{i\in I}\omega_i\cdot\lambda_{\rho(i)}=s$$

We can utilize this property to recover the secret. By contrast, if $S\notin\mathbb{A}$, the recovery algorithm can by no means recover s.

C. Complexity Assumption

Our model is based on the following complexity assumption:

Definition 3 (Discrete Logarithm Assumption): Given two elements $P,Q\in\mathbb{G}_1$, no probabilistic polynomial-time algorithm can find an integer $n\in Z_p^*$ with a non-negligible advantage for which the following holds:

$$Q=P^n$$

Definition 4 (Decisional Bilinear Diffie-Hellman Assumption): For an arbitrary group of exponents $a,b,c\in Z_p^*$, given a tuple (g,g^a,g^b,g^c,z), no probabilistic polynomial-time algorithm can distinguish the following two tuples with a non-negligible advantage:

$$(A=g^a,B=g^b,C=g^c,e(g,g)^{abc})$$
$$(A=g^a,B=g^b,C=g^c,e(g,g)^z)$$

IV. Model of CKM-CP-ABE for Cloud Data-Sharing

The model of our system is shown in Fig. 1. As depicted, five entities are involved in cloud data sharing:

1) **Client:** A client (CL) is a user who intends to access data in cloud storage via front-end devices. With the potential trend of mobile cloud services, mobile devices are the majority of front-end deices. If the CL's attribute set satisfies an access policy associated with ciphertext, the CL will be allowed to acquire plaintext. We assume that most mobile devices are performance-restrained, so CLs may be in danger of suffering key exposure.
2) **Key Authority:** The key authority (KA) is a vital component in the system. The KA is responsible for most calculating tasks, including key generation, key update, etc. We assume that the KA is semi-trusted in our system, meaning it is curious about the value of plaintext but has no intention of tampering with it.
3) **Cloud Server:** A cloud server (CS) is responsible for cloud storage management. All the data to be shared is in the control of the CS. We assume that any CS is semi-trusted.
4) **Decryption Server:** The decryption server (DS) has powerful computing capabilities. It undertakes and isolates the most, but not all task of decryption. We assume that the DS is semi-trusted and the DS access channel is insecure, because it is sufficient for CKM-CP-ABE to guarantee data security, which will be demonstrated in Section IV.
5) **Data Owner:** A data owner (DO) is an authorized user in the system who possesses data to be uploaded. DOs define their own explicit access policies so that only desirable CLs are granted permission to obtain plaintext.

We assume that all entities involved in data sharing does not collude with each other to access data illegally, otherwise the scheme will be unavailable and meaningless. In our model, attributes are authenticated by the KA. All granted attributes are represented by a group of random elements included in public parameters, which is generated by the KA in collaboration with a CS. Let *params* be public parameters. When a DO intends to share data, it encrypts the data using *params* sent to form the initial ciphertext CT_{init} and uploads it to the KA. The KA re-encrypts the initial ciphertext to form the ultimate ciphertext CT_{ulti}, which is sent to and stored in a CS. According to the CL's attribute set $S=\{\theta_1,\theta_2,\theta_3,\ldots\}$, the key management protocol helps to simultaneously and secretly generate three different components of the private key, namely, CPK_1, CPK_2 and CPK_3, each of which is kept by one of KA, CS or CL. Once asked for data stored in the cloud, the DS receives CPK_1 and CPK_2 to transform CT_{ulti} to CT'. Eventually, the CL extracts the plaintext $\mathcal{M}$ from CT' by its CPK_3.

For our proposed CKM-CP-ABE for cloud data-sharing, only by the combination of all three private key components can plaintext be extracted from the ultimate ciphertext. It means a CL requires collaboration with the KA and CS for decrypt ciphertext.

V. Main Key Management Algorithm

For clarity, we provide some notation from [2]: Define the universe of all system CLs as $U=\{u_1,u_2,\cdots,u_n\}$ and the universe of all granted attributes as $L=\{\theta_1,\theta_2,\cdots,\theta_m\}$. Let $G_t\subseteq U$ be an attribute group of CLs who share θ_t. Let $K_t\in Z_p^*$ be an attribute group key associated with G_t. Let $\mathcal{G}=\{G_t\mid\forall\theta_t\in L\}$ be the collection of all attribute groups and $\mathcal{K}=\{K_t\mid\forall\theta_t\in L\}$ be the collection of all attribute group keys. Our CKM-CP-ABE scheme consists of six algorithms whose details follow.

A. Setup

Let k be a security parameter. $\mathbb{G}_1$ and $\mathbb{G}_2$ are two multiplicative cyclic groups with prime order p. Let g be a generator of $\mathbb{G}_1$. Define two random oracles:

$$H:\{0,1\}^*\to\mathbb{G}_1$$
$$H_1:\mathbb{G}_2\to Z_p^*$$

The setup algorithm takes as input the security parameter k and the universe L. It returns a group of public parameters. It consists of the three following steps:

1) $TrustSetup(1^k,L)$: The trust launcher selects a group of random elements $h_1,h_2,\cdots,h_m\in_R\mathbb{G}_1$. For each $t\in\{1,2,\cdots,m\}$, the element h_t corresponds with an authorized attribute θ_t. The trust launcher then outputs the public parameter:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2707126, IEEE Access

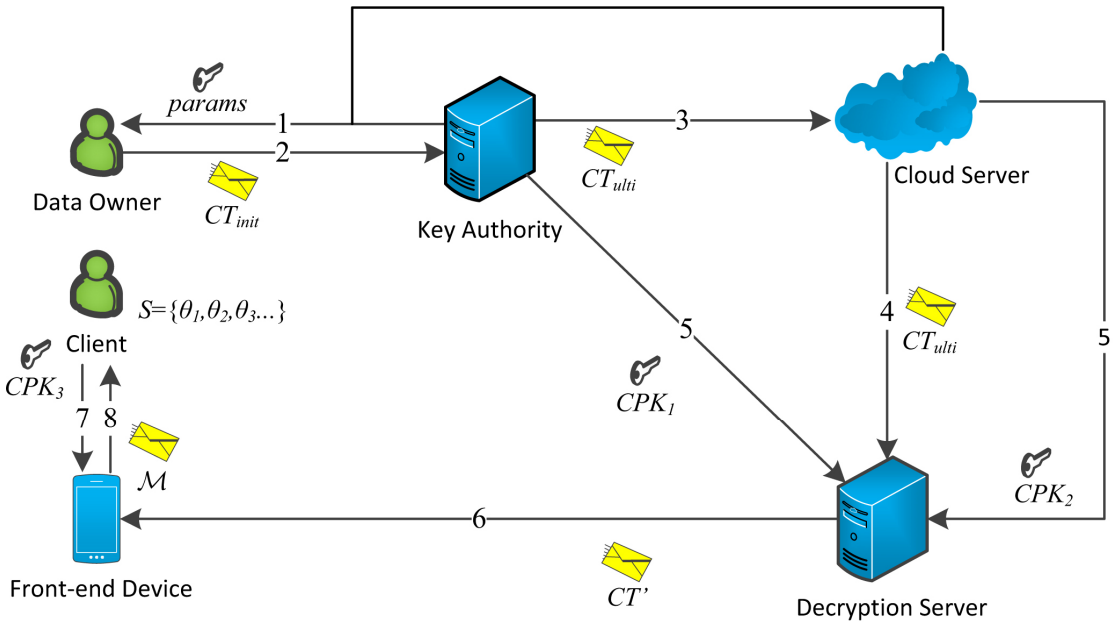

Fig. 1. The model of CKM-CP-ABE for cloud data sharing.

$$PP=\{g,h_1,h_2,\cdots,h_m,H,H_1\}$$

2) *KASetup*: The KA chooses a random exponent $q\in_R Z_p^*$ as a master secret and computes a public parameter g^q, by the following:

$$(MK_{ka}=q, PP_{ka}=g^q)$$

3) *CSSetup*: The CS chooses a random exponent $\alpha\in_R Z_p^*$. Then, it computes g^α and $e(g,g)^\alpha$ as a master secret/public parameter pair, that is:

$$(MK_{cs}=g^\alpha, PP_{cs}=e(g,g)^\alpha)$$

Let $params=(PP,PP_{ka},PP_{cs})$ be public parameters output by the setup algorithm. As mentioned in reference [10], PP is theoretically selected by a trust launcher before the initialization of the scheme. For our CKM-CP-ABE, we maintain the trust setup algorithm run by a trust launcher to keep theoretical correctness. In practical realization, since PP does not contain any sensitive knowledge, it is not very necessary to care about its generation for confidentiality. However, global consistency should be guaranteed by the appointment of the KA to generate and broadcast PP.

B. Key Generation

The key generation algorithm, which is run by the KA, takes as input the public parameter PP and a CL's attribute set S. Then, the KA chooses a random exponent $\tau\in_R Z_p^*$. Then, the key generation algorithm outputs the initial key as follows:

$$PK_{init}=(g^\tau,\forall x\in S: h_x^\tau)$$

The initial key is kept by the KA for subsequent private key updates.

C. Encryption

The encryption algorithm, which is run by the DO, takes as input the public parameters $params=(PP,PP_{ka},PP_{cs})$, an access structure $\mathbb{A}$ and plaintext $\mathcal{M}$. Meanwhile, the DO generates a random vector $\boldsymbol{v}=(s,y_2,y_3,\cdots,y_d)$ and a linear secret sharing scheme (M,ρ) associated with $\mathbb{A}$ to calculate $\lambda_i=\boldsymbol{v}\cdot M_i$ for each $u_i\in U$. Then, the DO outputs the initial ciphertext:

$$CT_{init}=((M,\rho),C=\mathcal{M}\cdot e(g,g)^{\alpha s},$$
$$C_\perp=g^s,\forall\theta_t\in\mathbb{A}: C_t^*=g^{q\lambda_t}h_{\rho(t)}^{-s})$$

D. Re-encryption

The re-encryption algorithm, which is run by the CS, takes as input the public parameters $params=(PP,PP_{ka},PP_{cs})$, the initial ciphertext CT_{init}, and the collection of attribute groups $\mathcal{G}$. We adopt the attribute group-based algorithm of [10] to re-encrypt the initial ciphertext. On receiving CT_{init} and $\mathcal{G}$, the CS selects two random exponents $\mu,\gamma\in_R Z_p^*$ and generates a set of attribute group keys $\mathcal{K}$. Each CL $u_k\in\mathcal{G}$ has a unique constant identity code, labeled $ID_k\in\{0,1\}^*$, regardless of attribute set changes. Subsequently, CS executes the following steps:

1) Compute $Q_k=H(ID_k)$ and the cloud server session key of cloud server $SK=g^\gamma$. Then, generate the element $x_k=H_1(e(Q_k^\mu,SK))$ of u_k.
2) For each $G_t\in\mathcal{G}$, construct a corresponding polynomial function $f_t(x)=\prod_{i=1}^{v}(x-x_i)=\sum_{i=0}^{v}a_ix^i(\mathrm{mod}\ p)$, where v represents the number of CLs in G_t.
3) Define the tuple $\{P_0,P_1,\cdots,P_v\}=\{g^{a_0},g^{a_1},\cdots,g^{a_v}\}$ and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2707126, IEEE Access

select a random $R \in_R Z_p^*$ to compute:

$$Hdr_t = (K_t \cdot P_0^R, P_1^R, \cdots, P_v^R)$$

4) Build a header message as follows:

$$Hdr = (g^{\mu}, \gamma, \forall G_t \in \mathcal{G} : Hdr_t)$$

5) Compute $H(ID_k)^{\gamma}$ as the session key of u_k.

6) Compute the ultimate ciphertext as follows:

$$CT = ((M, \rho), C = \mathcal{M} \cdot e(g,g)^{\alpha s},$$
$$C_{\perp} = g^s, \forall \theta_t \in \mathcal{A} : C_t = (g^{q\lambda_t} h_{\rho(t)}^{-s})^{K_t})$$
$$CT_{ulti} = (Hdr, CT)$$

The ultimate ciphertext is stored in the cloud. Once an access query is received, the CS will launch a retrieval. The goal of the re-encryption algorithm is realizing effective attribute revocation. More detail is provided in Section VI.

E. Private Key Update

The private key update algorithm, which is the principal innovation of CKM-CP-ABE, takes as input the parameters $params = (PP, PP_{ka}, PP_{cs})$ and the initial key PK_{init}. For this algorithm, the collaborative key management protocol is implemented to generate and distribute three different private key components. The protocol consists of two sub-protocols. The flows of the first sub-protocol are presented in Fig. 2.

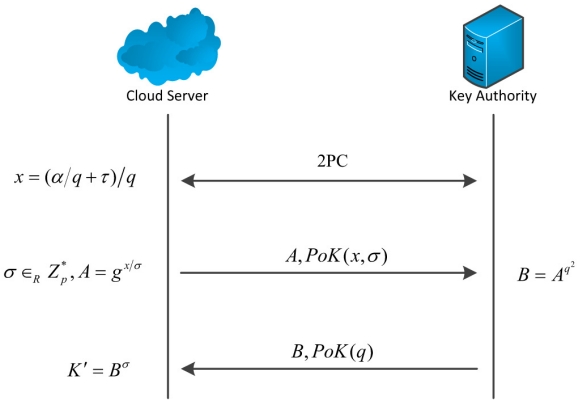

Fig. 2. The first sub-protocol flows.

Based on two party computation (2PC) [8][19], the first sub-protocol involves the KA and CS in the following simple arithmetic computation:

1) The KA chooses a consistent random exponent $\tau \in_R Z_p^*$.
2) The CS and KA engage in a secure 2PC, in which CS's input is α and KA's input is (q, τ). The 2PC returns a private output $x = (\alpha/q + \tau)/q$ to CS.
3) The CS chooses a random $\sigma \in_R Z_p^*$ and returns $A = g^{x/\sigma}$ to the KA.
4) The KA computes $B = A^{q^2}$ and sends it to the CS.
5) The CS obtains an initial key component by computing $K' = B^{\sigma} = g^{\alpha + q\tau}$.

As the depicted interactions in Fig. 2, $PoK(\cdot)$ represents a proof of knowledge of the secret, used in calculations. Here, we have omitted the statement being proved for simplicity. Schnorr protocol can be used to prove knowledge about statements efficiently.

Theorem 1. *The above computation for generating private key component $g^{\alpha+q\tau}$ is a secure protocol, assuming the underlying arithmetic 2PC and zero knowledge proofs are secure.*

Proof. We provide the proof of theorem 1 in Section VI.

Note that after the above process, the CS holds K' and the KA possesses the initial key PK_{init}. These two facts are crucial to the proposed second sub-protocol that involves the KA, CS, and CL. We present flows of the second sub-protocol in Fig. 3.

As depicted in Fig. 3, the computation is as follows:

1) The CS randomly selects $r_1, \pi_1 \in_R Z_p^*$ as its private input, and the KA randomly selects $r_2, \pi_2 \in_R Z_p^*$ as its private input.
2) The protocol returns private output $y = (r_1 + r_2)\pi_1\pi_2$ to both KA and CS.
3) The CS selects random exponent $\xi \in_R Z_p^*$ and returns $X_1 = (K')^{y/\xi}$ to the KA.
4) The KA returns $Y_1 = X_1^{1/\pi_2^2}$ to the CS.
5) The CS outputs $K = Y_1^{\xi/\pi_1^2}$ as its component of the private key.
6) The KA selects random exponent $\varsigma \in_R Z_p^*$ and returns $X_2 = (g^{\tau})^{y/\varsigma}$ and $\forall x \in S : X_x = (h_x^{\tau})^{y/\varsigma}$ to the CS.
7) The CS returns $Y_2 = X_2^{1/\pi_1^2}$ and $\forall x \in S : Y_x = X_x^{1/\pi_1^2}$ to the KA.
8) The KA outputs $D = Y_2^{\varsigma/\pi_2^2}$ and $\forall x \in S : D_x = Y_x^{\varsigma/\pi_2^2}$.
9) The protocol secretly sends $(r_1 + r_2)/\pi_1\pi_2$ to the CL as its component of the private key.

Thus, the components of the private key are:

$$CPK_1 = (K = (g^{\alpha+q\tau})^{(r_1+r_2)/\pi_1\pi_2})$$
$$CPK_2 = (D = (g^{\tau})^{(r_1+r_2)/\pi_1\pi_2}, \forall x \in S : D_x = (h_x^{\tau})^{(r_1+r_2)/\pi_1\pi_2})$$
$$CPK_3 = (r_1 + r_2)/\pi_1\pi_2$$

As shown above, CPK_1 and CPK_2 are, respectively, kept by the KA and CS, while CPK_3 is held by the CL. All private key component generation and transmission is executed under the collaborative key management protocol. Neither KA nor CL can decrypt ciphertext on their own due to the proposed collaborative key management protocol. Thus, not only does our scheme avoid the key escrow problem, but it also helps prevent the key exposure.

Theorem 2. The second sub-protocol for generating all private key components is a secure protocol, assuming the first sub-protocol is secure.

Proof. The proof of theorem 2 is provided in Section VI. This proof is based on the proof of theorem 1.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2707126, IEEE Access

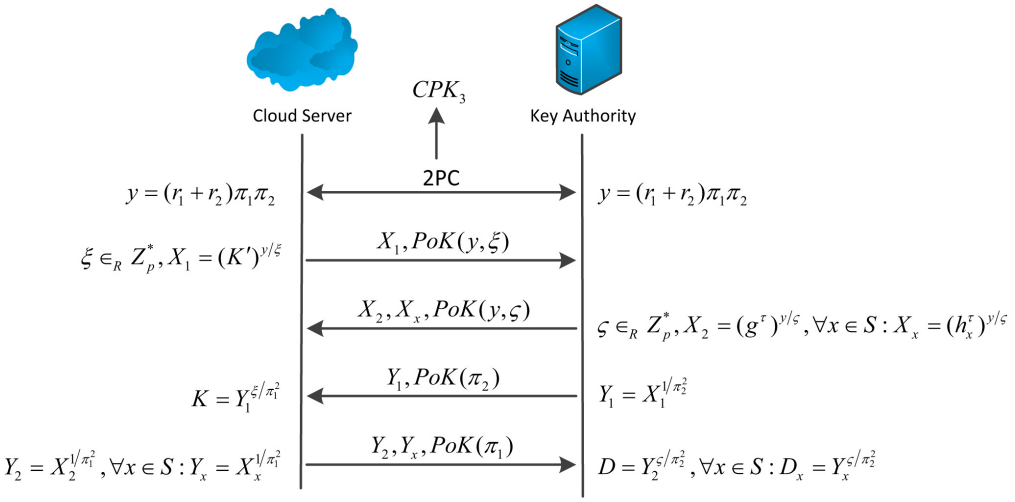

Fig. 3. The second sub-protocol flows.

F. Decryption

The decryption algorithm takes as input CL's attribute set S, the ultimate ciphertext CT_{ulti}, and all private key components (CPK_1, CPK_2, CPK_3). On receiving a decryption query from a CL u_k, the KA and CS send their private key components to the DS. Then, the CS helps the DS retrieve the session key of u_k and the ultimate ciphertext CT_{ulti} from cloud storage. For each attribute group key in the set $\{K_t \mid \forall \theta_t \in S\}$, the following computation is executed:

$$H_1(e(g^{\mu}, H(ID_k)^{\gamma})) = x_k$$

$$K_t \cdot P_0^R \prod_{i=1}^{v} P_i^{R \cdot x_k^j} = K_t$$

After obtaining K_t, the DS extracts an element T_0 as follows:

$$\begin{aligned}
&\prod_{\rho(t)\in S}(e(C_t^{1/K_t}, D)\cdot e(C_{\perp}, D_{\rho(t)}))^{\omega_t}\\
&=\prod_{\rho(t)\in S}(e((g^{q\lambda_t}h_{\rho(t)}^{-s})^{K_t\cdot 1/K_t}, (g^{\tau})^{(r_1+r_2)/\pi_1\pi_2})\cdot e(g^s, (h_{\rho(t)}^{r})^{(r_1+r_2)/\pi_1\pi_2}))^{\omega_t}\\
&=\prod_{\rho(t)\in S}(e(g,g)^{q\lambda_t\tau(r_1+r_2)/\pi_1\pi_2})^{\omega_t}\\
&=e(g,g)^{qs\tau(r_1+r_2)/\pi_1\pi_2}\\
&=T_0
\end{aligned}$$

Next, it extracts another element T_1 as follows:

$$\begin{aligned}
&e(C_{\perp}, K)/T_0\\
&=e(g^s, (g^{\alpha+q\tau})^{(r_1+r_2)/\pi_1\pi_2})/e(g,g)^{qs\tau(r_1+r_2)/\pi_1\pi_2}\\
&=e(g,g)^{\alpha s(r_1+r_2)/\pi_1\pi_2}\\
&=T_1
\end{aligned}$$

After that, it sends T_1 to u_k. Then, u_k obtains plaintext via the following easy computation:

$$\begin{aligned}
&C/T_1^{\pi_1\pi_2/(r_1+r_2)}\\
&=\mathcal{M}\cdot e(g,g)^{\alpha s}/(e(g,g)^{\alpha s(r_1+r_2)/\pi_1\pi_2})^{\pi_1\pi_2/(r_1+r_2)}\\
&=\mathcal{M}
\end{aligned}$$

VI. Comprehensive Analysis

TABLE I
Notations Relevant to Efficiency Comparison

Symbol	Definition
$\lvert\mathbb{G}_1\rvert$	Size of an element in $\mathbb{G}_1$
$\lvert\mathbb{G}_2\rvert$	Size of an element in $\mathbb{G}_2$
$\lvert\mathbb{A}\rvert$	Size of an access structure
$\lvert Z_p^*\rvert$	Size of an element in Z_p^*
m	Sum of all authorized attribute in system
Σ	Sum of attributes in an access structure
$\lvert S\rvert$	Sum of attributes held by a CL
v	Sum of CLs in an attribute group
$ST_{\wedge}$	Sum of threshold value of all "AND" gates in an access tree
SG	Sum of gates in an access tree
C_p	A calculation of pairing
C_h	A calculation of hash function
C_e	A calculation of exponentiation
C_m	A calculation of multiplication
C_d	A calculation of division

In our analysis, we use notation shown in Table. I.

A. Security Requirements

1) Data confidentiality

For the CKM-CP-ABE model, we assume that all entities involved in data sharing are semi-trusted but do not collude with each other to access data illegally. Since the semi-trusted KA and CS are honest but curious about plaintext, the shared data should be kept secret from them as well as from unauthorized CLs.

In our scheme, the KA receives the initial ciphertext from the DO and generates the initial key corresponding to the attribute set of the CL, similar to what the key generation center should do in [2] and [5]. However, the KA cannot decrypt the initial ciphertext using the initial key, since other indispensable secrets are kept by the CS that the KA does not know. Thus, it is impossible for the KA to obtain plaintext by itself. Similarly,

the CS is also unable to extract data from ciphertext on its own.

The DS transforms the ultimate ciphertext to semi-decrypted ciphertext, which is constant-size and shorter than the ultimate ciphertext. The semi-decrypted ciphertext is considered to be ElGamal encrypted. During the transformation, the DS cannot acquire any knowledge about the data. Thus, data security is guaranteed although the DS is semi-trusted and the DS access channel is insecure.

Furthermore, it is trivially guaranteed that outside visitors have no access to plaintext. Any CL that does not hold authorized attributes cannot extract adequate attribute group keys to obtain T_0 and T_1. Accordingly, such a CL cannot recover the value of $e(g,g)^{\alpha s}$. Consequently, decryption does not work correctly with fewer than three distinct private key components.

2) Backward and forward secrecy

The first ABE key revocation mechanism was proposed by Bethencourt *et al.* [2]. A timed mechanism associates a validation time with each attribute [28]. It is impractical, however, to attach the exact expiration time to each attribute of each CL. Additionally, the timed mechanism fails to guarantee backward and forward secrecy [22]. Green *et al.* [5] mainly concentrated on decryption efficiency while ignoring revocation block construction. Based on Bethencourt et al.'s process [2], Hur [10] and Xiong et al. [17] adopted an attribute group mechanism to realize immediate revocation. For CKM-CP-ABE, we introduce a scheme inspired by [10] to guarantee backward and forward secrecy.

Once a CL obtains a new attribute, the KA updates the corresponding attribute group key. Then, it embeds the newly rebuilt head message into the ciphertext. Finally, all CLs in the new attribute group obtain new private key components. Even if this CL possesses ciphertext encrypted before acquiring this attribute, it cannot be correctly decrypted by this CL. Thus, backward secrecy can be guaranteed.

If some CLs lose an attribute, the KA will update the attribute group list at once. Then, the cloud server immediately updates the corresponding set of attribute group keys for re-encrypting ciphertext by rebuilding the head message. Later, the collaborative key management protocol distributes updated private key components. For ciphertext accessible only to those who hold this attribute, those who no longer possess this attribute can by no means extract any information from the ciphertext. Thus, forward secrecy can be guaranteed as well.

3) Collusion resistance

In ABE, when unauthorized users succeed in recovering the master secret via their collusion, they can extract the plaintext. Taking the one-to-many property into account, it is necessary for ABEs to construct an effective mechanism against a collusion attack. Current ABEs can trivially deter unauthorized users from illegal collusion to gain knowledge by embedding a randomized secret into the private key of each user [7][18][21][29]. In CKM-CP-ABE, each private key is randomized by inserting a unique random τ. Even if CLs combine their keys, they still cannot recover $e(g,g)^{\alpha s}$. Therefore, the proposed scheme can inherently defend against collusion attack.

4) Key escrow free

In most ABEs (e.g. [2][3][3][5][6][7][29]), the key authority must be fully trusted, since it has absolute power to generate the private keys of all users. Accordingly, the KA can also decrypt all the ciphertext by using generated private keys without the users' permission. This is the so-called key escrow problem, which is an inherent flaw threatening privacy of global users. Inspired by [8] and [10], we consider the KA semi-trusted, which means it tries to extract the value of plaintext but has no malicious intent. In the CKM-CP-ABE, since α and (q,τ) are, respectively, kept by the KA and CS in secret, it is impossible for the KA alone to extract any information from ciphertext. Consequently, our CKM-CP-ABE perfectly address the key escrow problem.

5) Key exposure free

The use of cloud storage services for easy backup and access across multiple devices is getting popular with the increase in mobile devices on the Internet [24]. Nevertheless, this trend is not accompanied by a comparable increase in mobile security protection. Mobile devices, such as smartphones, are far inferior to cloud storage servers in privacy protection [20], which may lead to the exposure of users' private keys stored in these devices. In the CKM-CP-ABE model, a CL's private key consists of three parts, generated and held, respectively, by the KA, CS, and CL. Possession of anything less than all three of these components causes decryption failure. Thus, it effectively addresses key exposure. Noting that among [2][5][10] and the proposed CKM-CP-ABE, only CKM-CP-ABE can prevent key exposure.

B. Revocation

When a CL loses an attribute, e.g., θ_l, the proposed CKM-CP-ABE will immediately execute the revoking operation. At the beginning of revocation, the KA updates the attribute group from $\mathcal{G}$ to $\mathcal{G}'$. The CS runs the re-encryption algorithm upon receiving $(CT_{init}, \mathcal{G}')$. The CS promptly updates the attribute group key of G_l:

$$K_l \to K_l'$$

The updated attribute group key set is

$$\mathcal{K}' = \{K_l', \forall \theta_t \in L \setminus \{\theta_t\} : K_t\}$$

Let v' represent the number of CLs in G_l. For G_l, the reconstructed polynomial function is

$$f_l(x) = \prod_{i=1}^{v'} (x - x_i) = \sum_{i=0}^{v'} a_i' x^i \pmod p$$

Subsequently, re-encryption selects a new random $R' \in_R Z_p^*$ to build a new header message as follows:

$$\{P_0, P_1, \cdots, P_{v'}\} = \{g^{a_0'}, g^{a_1'}, \cdots, g^{a_{v'}'}\}$$

$$Hdr_l = \{K_l' \cdot P_0^{R'}, P_1^{R'}, \cdots, P_v^{R'}\}$$

$$Hdr' = \{g^{\mu}, Hdr_l', \forall G_t \in \mathcal{G} \setminus \{G_l\} : Hdr_t\}$$

Next, the ultimate ciphertext is updated:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2707126, IEEE Access

$$CT' = ((M,\rho), C = \mathcal{M}\cdot e(g,g)^{\alpha s}, C_{\perp} = g^{s}, C_{l} = (g^{q\lambda_l} h_{\rho(l)}^{-s})^{K_l'},$$
$$\forall \theta_t \in L \setminus \{\theta_l\} : C_t = (g^{q\lambda_t} h_{\rho(l)}^{-s})^{K_t})$$
$$CT'_{ulti} = (Hdr, CT')$$

Finally, the second sub-protocol is implemented to update the private key components:

$$CPK_1' = (K = (g^{\alpha+q\tau})^{(r_1'+r_2')/\pi_2'\pi_2'})$$
$$CPK_2' = (D = (g^{\tau})^{(r_1'+r_2')/\pi_2'\pi_2'}, D_l = (h_l^{\tau})^{(r_1'+r_2')/\pi_2'\pi_2'},$$
$$\forall x \in S \setminus \{l\} : D_x = (h_x^{\tau})^{(r_1'+r_2')/\pi_2'\pi_2'})$$
$$CPK_3 = (r_1'+r_2')/\pi_2'\pi_2'$$

C. Efficiency

In terms of efficiency, we compare the proposed CKM-CP-ABE with other representative ABE variants detailed in methods of Bethencourt *et al.* [2], Green *et al.* [5], and Hur [10]. We summarize the efficiency comparison with respect to public key, ciphertext, and private key size in Table. II. For CKM-CP-ABE, the public key size is $(\Sigma+1)|\mathbb{G}_1|+|\mathbb{G}_2|$, which is larger than those of other schemes because our key requires a tuple of random elements $h_1, h_2, \cdots, h_m$ respectively associated with each authorized attribute. The CKM-CP-ABE ciphertext size is $(\Sigma+1)|\mathbb{G}_1|+|\mathbb{G}_2|+|\mathbb{A}|$, which is the smallest among the compared schemes. Similar to Green *et al.*'s [2] scheme, the private key size of CKM-CP-ABE is $|Z_p^*|+(|S|+2)|\mathbb{G}_1|$, in which the number of elements in $\mathbb{G}_1$ is less than in the methods of Bethencourt *et al.* [2] and Hur [10].

The comparison of total decryption overhead and client decryption overhead for each method is presented in Table. III. Enormous computation overhead is known to be an ABE bottleneck. Mass calculations of pairings are the main contributors to this computation load. Similar to other schemes [2][5][10], CKM-CP-ABE has heavy total decryption overhead. Introducing the attribute group mechanism adds more exponentiation and multiplication calculation to our scheme. Due to the collaborative mechanism, however, we decrease client decryption overhead dramatically. Authorized CLs in our scheme require only one exponentiation calculation and one division calculation because the DS undertakes enormous computation without any knowledge leakage. Differed from [5], CKM-CP-ABE not only outsources decryption but also provides a collaborative mechanism with reliable security.

D. Security Proof

1) Proof of Theorem 1

We demonstrate the security of the first sub-protocol by examining the cases of a corrupt CS and corrupt KA separately.

For a corrupt CS, the simulator Sim_{ka} proceeds as follows: The arithmetic 2PC extracts α from CS in preparation for obtaining $x = (\alpha/q+\tau)/q$. Then, a random $x \in_R Z_p^*$ is selected and sent to the arithmetic 2PC simulator. Note that it is correctly distributed, since there exists an q such that $x = (\alpha/q+\tau)/q$ for any x, α, and τ. Then, the random x is transmitted to the adversary. Next, Sim_{ka} receives A from the corrupt CS with the corresponding zero knowledge proof. After that, we extract σ from the proof system with the extractor and send a random $B \in_R \mathbb{G}_1$ to the corrupt CS. Finally, α is sent to the trusted party for computation of K', which is returned to the KA.

Now consider a hybrid simulator Hyb_{ka} that takes as input q and τ from the KA. It computes $x = (\alpha/q+\tau)/q$ using the arithmetic 2PC simulator as described above. If $(\alpha/q+\tau)/q$

TABLE II
COMPARISON OF SIZE OF PUBLIC KEY, PRIVATE KEY, AND CIPHERTEXT

System	Public key size	Ciphertext size	Private key size
[2]	$\|\mathbb{G}_1\|+\|\mathbb{G}_2\|$	$(2\Sigma+1)\|\mathbb{G}_1\|+\|\mathbb{G}_2\|+\|\mathbb{G}_3\|$	$(2\|S\|+1)\|\mathbb{G}_1\|$
[5]	$\|\mathbb{G}_1\|+\|\mathbb{G}_2\|$	$(2\Sigma+1)\|\mathbb{G}_1\|+\|\mathbb{G}_2\|+\|\mathbb{G}_3\|$	$\|Z_p^*\|+(\|S\|+2)\|\mathbb{G}_1\|$
[10]	$\|\mathbb{G}_1\|+\|\mathbb{G}_2\|$	$(2\Sigma+1)\|\mathbb{G}_1\|+\|\mathbb{G}_2\|+\|\mathbb{G}_3\|$	$(2\|S\|+2)\|\mathbb{G}_1\|$
CKM-CP-ABE	$(\Sigma+1)\|\mathbb{G}_1\|+\|\mathbb{G}_2\|$	$(\Sigma+1)\|\mathbb{G}_1\|+\|\mathbb{G}_2\|+\|\mathbb{G}_3\|$	$\|Z_p^*\|+(\|S\|+2)\|\mathbb{G}_1\|$

TABLE III
COMPARISON OF DECRYPTION OVERHEAD

System	Decryption overhead	C_h	C_p	C_e	C_m	C_d
[2]	Total	0	$2\Sigma+1$	$ST_\wedge$	$ST_\wedge - SG$	2
	Client	0	$2\Sigma+1$	$ST_\wedge$	$ST_\wedge - SG$	2
[5]	Total	0	$\|S\|+2$	$2\|S\|+1$	$2\|S\|-1$	2
	Client	0	0	1	0	1
[10]	Total	1	$2\|S\|+2$	$ST_\wedge + v$	$ST_\wedge - SG + v$	2
	Client	1	$2\|S\|+2$	$ST_\wedge + v$	$ST_\wedge - SG + v$	2
CKM-CP-ABE	Total	1	$2\|S\|+2$	$2\|S\|+v+1$	$2\|S\|+v-1$	2
	Client	0	0	1	0	1

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2707126, IEEE Access

corresponding to α is correctly computed, the 2PC simulator runs as the real 2PC does. In the real protocol, q and σ are randomly picked so that x is uniformly distributed over Z_p^* and B is uniformly distributed over $\mathbb{G}_1$. Thus, the x and B generated in Sim_{ka} are distributed identically to those in Hyb_{ka}. Assuming security of the proof of knowledge is guaranteed, Hyb_{ka} should be indistinguishable from Sim_{ka}.

For a corrupt KA, the simulator Sim_{cs} proceeds as follows: The arithmetic 2PC simulator calculates $(\alpha/q+\tau)/q$ and extracts τ. Then, a random value $A \in_R \mathbb{G}_1$ is chosen and sent to the corrupt KA. When Sim_{cs} receives B from the adversary, the arithmetic 2PC extracts and sends q to the trusted party, which computes $K' = g^{\alpha+q\tau}$. Finally, K' is transferred to the CS as private output.

Now consider a hybrid simulator Hyb_{cs} that takes as input the secret α. It first runs the arithmetic 2PC simulator to compute x. If this protocol provides the correct output value of q and τ, the simulator executes all steps as in the real protocol. This is totally indistinguishable from the real process aided by arithmetic 2PC security. Regardless of whether the real protocol or the simulator is used, A is distributed uniformly over $\mathbb{G}_1$ since σ is chosen at random. Thus, the values of x and A generated by Sim_{cs} are distributed identically to those in Hyb_{cs}. Assuming the proof of knowledge is secure, Hyb_{cs} should be indistinguishable from Sim_{cs}.

Thus, we conclude that our first sub-protocol is secure.

2) Proof of Theorem 2

To prove theorem 2 efficiently, we need to modify the second sub-protocol slightly to build a simulator. If we assume that the CS and KA both obtain the secret y via 2PC but do not ascertain what secret the other entity obtains, then neither has the advantage of guessing the other's secret knowledge. Accordingly, the two opposing directions of secret sharing can be regarded as independent. We slightly modify the second sub-protocol by splitting it into two independent phases. We define the simulator Sim_{sec} of the second sub-protocol as presented in Fig. 4.

As is depicted, both phase one and phase two can be viewed as analogous to the first sub-protocol. Phase one receives random exponents $r_1, r_2, \pi_1, \pi_2 \in_R Z_p^*$ from the CS and KA, then executes a secure interaction for computing $K = (g^{\alpha+q\tau})^{(r_1+r_2)/\pi_1\pi_2}$. Phase two receives random exponents $b_1, b_2, \omega_1, \omega_2 \in_R Z_p^*$ from the CS and KA, then executes a secure interaction to compute $D = (g^{\tau})^{(b_1+b_2)/\omega_1\omega_2}$ and $\forall x \in S : D_x = (h_x^{\tau})^{(b_1+b_2)/\omega_1\omega_2}$.

For phase one, the simulator Sim_1 proceeds as in the proof of theorem 1 for the cases of corrupt CS and KA. Now consider the hybrid simulator Hyb_1. If this protocol provides correct outputs, the all steps are completely executed as in the real protocol. So, the hybrid simulator is totally indistinguishable from the real one aided by arithmetic 2PC security.

For phase two, the simulator Sim_2 also proceeds as in the proof of theorem 1 for the cases of corrupt CS and KA. For the hybrid simulator Hyb_2, if this protocol provides correct output values, it completely executes all steps as in the real protocol. So, the hybrid simulator is totally indistinguishable from the real one aided by arithmetic 2PC security.

Based on this analysis, we consider the hybrid simulator Hyb_{sec}. When 2PC provides the private secret of both entities, the hybrid simulator correctly computes the value of y and then executes exactly as the real protocol does. Regardless of which entity is corrupted, Hyb_{sec} will complete execution as Sim_{sec} does. We note that it is clearly indistinguishable from the real protocol because of the independence of the secret

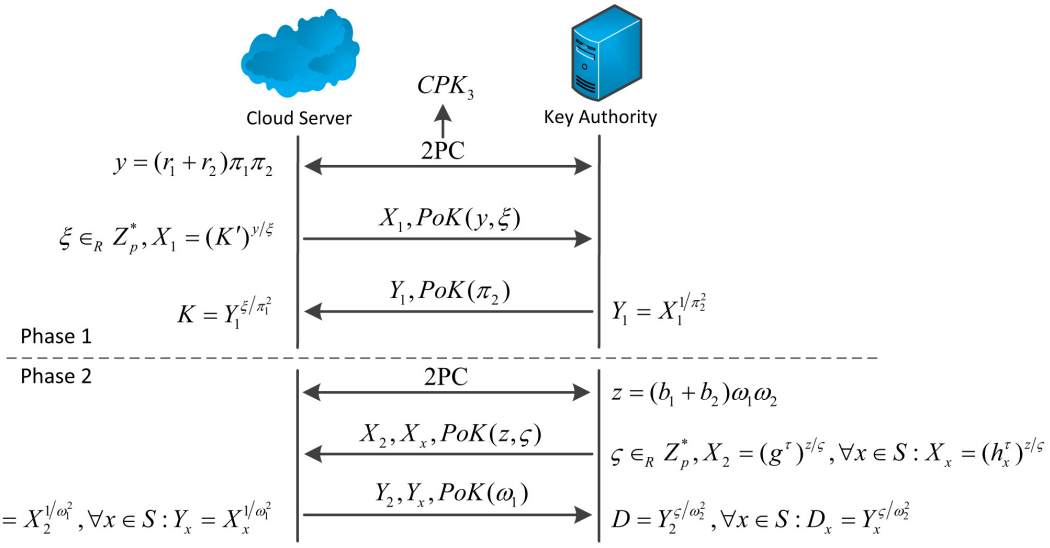

Fig. 4. The simulator of the second sub-protocol flows.

sharing's two directions and the uniform distribution of random exponents in the real protocol.

Thus, the second sub-protocol is theoretically secure.

VII. Conclusion and Future Work

Ciphertext policy attribute-based encryption is a promising cryptographic technique to realize fine-grained access control in secure cloud storage. In this paper, we propose a novel collaborative key management protocol to enhance both security and efficiency of key management in ciphertext policy attribute-based encryption for cloud data sharing system. Distributed key generation, issue and storage of private keys are realized without adding any extra physical infrastructure. We introduce attribute groups to build a private key update algorithm for fine-grained and immediate attribute revocation. The proposed collaborative mechanism perfectly addresses not only key escrow problem but also a worse problem called key exposure that previous research hardly noticed. Meanwhile it helps to optimize clients' user experience since only a small amount of responsibility is taken by them for decryption. Thus, the proposed scheme performs better in cloud data sharing system serving massive performance-restrained front-end devices with respect to either security or efficiency.

Our future work will build on the preliminary findings in this work to develop the proposed scheme by reducing ciphertext size, encryption cost and decryption cost, which are still open problems that hinder practical application of attribute-data sharing. Considering some specific industrial scenarios such as personal health record access control, besides, the expressiveness of access policy needs enhancement as well.

References

[1] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proc. EuroCrypt*, 2005, pp. 457-473.

[2] J. Bethencourt, A. Sahai, and B.Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Secur. Privacy*, 2007, pp. 321-334.

[3] N. Attrapadung and H. Imai, "Conjunctive broadcast and attribute-based encryption," in *Proc. Int. Conf. Pairing-Based Cryptography*, 2009, pp. 248-265.

[4] B. Waters, "Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization," in *Proc. Public Key Cryptography*, 2011, pp. 53-70.

[5] M. Green, S. Hohnberger, and B. Waters, "Outsourcing the decryption of ABE ciphertext," in *Proc. USENIX Secur. Symp.*, 2011, pp. 34.

[6] J. Lai, R. H. Deng, C. Guan, and J. Weng, "Attribute-based encryption with verifiable outsourced decryption," *IEEE Trans. Inf. Forens. Security*, vol. 8, no. 8, pp. 1343-1354, 2013.

[7] S. Lin, R. Zhang, H. Ma, and M. Wang, "Revisiting attribute-based encryption with verifiable outsourced decryption," *IEEE Trans. Inf. Forens. Security*, vol. 10, no. 10, pp. 2119-2130, 2015.

[8] M. Chase, and S. S. M. Chow, "Improving privacy and security in multi-authority attribute-based encryption," in *Proc. ACM CCS*, 2009, 121-130.

[9] G. Zhang, L. Liu, and Y. Liu, "An attribute-based encryption scheme secure against malicious KGC," in *Proc. TRUSTCOM*, 2012, pp. 1376-1380.

[10] J. Hur, "Improving security and efficiency in attribute-based data sharing," *IEEE Trans. Knowl. Data. Eng.*, vol. 25, no. 10, pp. 2271-2282, 2013.

[11] P. P. Chandar, D. Mutkurman, and M. Rathinrai, "Hierarchical attribute based proxy reencryption access control in cloud computing," in *Proc. ICCPCT*, 2014, pp. 1565-1570.

[12] X. A. Wang, J. Ma, and F. Xhafa, "Outsourcing decryption of attribute based encryption with energy efficiency," in *Proc. 3PGCIC*, 2015, pp. 444-448.

[13] L. Cheung, and C. Newport, "Provably secure ciphertext policy ABE," in *Proc. ACM CCS*, 2007, pp. 456-465.

[14] J. Hur, and D. K. Noh, "Attribute-based access control with efficient revocation in data outsourcing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 7, pp. 1214-1221, 2011.

[15] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters, "Secure attribute-based systems," in *Proc. ACM CCS*, 2006, pp. 99-112.

[16] A. Boldyreva, V. Goyal, and V. Kumar, "Identity-based encryption with efficient revocation," in *Proc. ACM CCS*, 2008, pp. 417-426.

[17] A. Xiong, C. Xu, and Q. Gan, "A CP-ABE scheme with system attributes revocation in cloud storage," in *Proc. ICCWAMIP*, 2014, pp. 331-335.

[18] Q. Wu, "A generic construction of ciphertext-policy attribute-based encryption supporting attribute revocation," *China Commun.*, vol. 11, no. 13, pp. 93-100, 2014.

[19] S. S. M. Chow, "Removing escrow from identity-based encryption," in *Proc. Int. Conf. Practice and Theory in Public Key Cryptography*, 2009, pp. 256-276.

[20] M. S. Ahmad, N. E. Musa, R. Nadarajah, R. Hassan, and N. E. Othman, "Comparison between android and iOS operating system in terms of security," in *Proc. CITA*, 2013, pp. 1-4.

[21] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. ACM CCS*, 2006, pp. 89-98.

[22] S. Rafaeli, and D. Hutchison, "A survey of key management for secure group communication," *ACM Computing Survey*, vol. 35, no. 3, pp. 309-329, 2003.

[23] S. Subashini, and V. Kavitha, "A survey on security issues in service delivery models of cloud computing," *Journal of Network and Computer Applications*, vol. 34, no. 1, pp. 1-11, 2011.

[24] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless Communications and Mobile Computing*, vol. 13, no. 18, pp. 1587-1611.

[25] H. Takabi, J. B. D. Joshi, and G. J. Ahn, "Security and privacy challenges in cloud computing enviroments," IEEE Security & Privacy, vol. 8, no. 6, pp. 24-31, 2010.

[26] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for secure cloud storage," *IEEE Trans. Comput.*, vol. 62, no. 2, pp. 362-375, 2013.

[27] 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 Trans. Parallel Distrib. Syst.*, vol. 24, no. 1, pp. 131-143, 2013.

[28] Q. Liu, G. Wang, and J. Wu, "Time-based proxy re-encryption scheme for secure data sharing in cloud environment," *Information Sciences*, vol. 258, pp. 355-370, 2014.

[29] X. Yao, Z. Chen, and Y. Tian, "A lightweight attribute-based encryption scheme for the Internet of Things," *Future Generation Computer Systems*, vol. 49, pp. 104-112, 2015.

[30] H. Hong, Z. Sun, "High efficient key-insulated attribute based encryption scheme without bilinear pairing operation," *SpringerPlus*, vol. 5, no. 1, pp. 131, 2016.

[31] M. R. Rahimi, J. Ren, C. H. Liu, A. V. Vasilakos, and N. Venkatasubramanian, "Mobile cloud computing: a survey, state of art and future directions," *Mobile Networks and Applications*, vol. 19, no. 2, pp. 133-143, 2014.

[32] D. Pletea, S. Sedghi, M. Veeningen, and M. Petkovic, "Secure distributed key generation in attribute based encryption systems," in *Proc. ICITST*, 2015, pp. 103-107.

[33] X. L. Xu, J. L. Zhou, X. H. Wang, and Y. Zhang, "Multi-authority proxy re-encryption based on CPABE for cloud storage systems," *Journal of Systems Engineering and Electronics*, vol. 27, no. 1, pp. 211-223, 2016.

[34] S. Easwarmoorthy, F. Sophia, and A. Karrothu, "An efficient key management infrastructure for personal health records in cloud," in *Proc. WiSPNET*, 2016, pp. 1651-1657.

[35] N. Oualha, and K. T. Nguyen, "Lightweight attribute-based encryption for the Internet of Things," in *Proc. ICCCN*, 2016, pp. 1-6.