2-2018

Secure fine-grained access control and data sharing for dynamic groups in the cloud

Shengmin XU

Guomin YANG

Yi MU

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

DOI: https://doi.org/10.1109/TIFS.2018.2810065

Secure Fine-Grained Access Control and Data Sharing for Dynamic Groups in the Cloud

Shengmin Xu, Guomin Yang, *Senior Member, IEEE*, Yi Mu, *Senior Member, IEEE*, and Robert H. Deng, *Fellow, IEEE*

***Abstract*— Cloud computing is an emerging computing paradigm that enables users to store their data in a cloud server to enjoy scalable and on-demand services. Nevertheless, it also brings many security issues, since cloud service providers (CSPs) are not in the same trusted domain as users. To protect data privacy against untrusted CSPs, existing solutions apply cryptographic methods (e.g., encryption mechanisms) and provide decryption keys only to authorized users. However, sharing cloud data among authorized users at a fine-grained level is still a challenging issue, especially when dealing with dynamic user groups. In this paper, we propose a secure and efficient fine-grained access control and data sharing scheme for dynamic user groups by: 1) defining and enforcing access policies based on the attributes of the data; 2) permitting the key generation center to efficiently update user credentials for dynamic user groups; and 3) allowing some expensive computation tasks to be performed by untrusted CSPs without requiring any delegation key. Specifically, we first design an efficient revocable attribute-based encryption (ABE) scheme with the property of ciphertext delegation by exploiting and uniquely combining techniques of identity-based encryption, ABE, subset-cover framework, and ciphertext encoding mechanism. We then present a fine-grained access control and data sharing system for on-demand services with dynamic user groups in the cloud. The experimental data show that our proposed scheme is more efficient and scalable than the state-of-the-art solution.**

***Index Terms*— Cloud storage, data sharing, access control, revocation, dynamic group.**

I. Introduction

CLOUD computing is widely accepted as a new computing paradigm due to its intrinsic resource-sharing and low maintenance characteristics. In cloud computing, the CSPs, such as Amazons EC2 and S3, Google App Engine, and Microsoft Azure, are able to deliver various services, including software as a service (SaaS), platform as a service (PaaS) and infrastructure as a service (IaaS), to cloud users. By migrating the local data management system into cloud storage, users can enjoy cost savings and productivity enhancements by using cloud-based services to manage projects and establish collaborations. With the increasing development of cloud computing technologies, it is not hard to imagine that in the near future more and more businesses will be moved into the cloud.

One of the most fundamental services offered by CSPs is data storage. Despite of the benefits provided by cloud storage, it is facing many challenges that, if not well resolved, may impede its fast growth. Consider a practical application that a company allows its staff or departments to store and share data via the cloud. By utilizing the cloud, the company can be completely released from the local data storage and maintenance burden. However, it also incurs a major security threat towards the data confidentiality. Specifically, the CSPs are not fully trusted by users while the data files stored in the cloud may be sensitive and confidential. To address this issue, a basic solution is to encrypt data, and then uploads the encrypted data into the cloud. However, the traditional encryption mechanisms are not efficient or flexible for data sharing in the cloud. In order to achieve optimal usage of storage resources, it is desirable to use advanced encryption mechanisms allowing the data to be shared at a fine-grained level. One of the promising tools for achieving fine-grained access control and sharing of encrypted data is to use attribute-based encryption (ABE) [1], [2]. Nevertheless, it is not straightforward to directly apply ABE in real applications due to various practicality concerns.

Dynamic User Groups: Dynamic user groups are very common in cloud applications, e.g., due to expiration or change of user membership and user credentials being stolen/compromised/misused. In dynamic user groups, user revocation is a critical security issue that must be properly addressed. However, one challenging problem in handling user revocation in cloud storage is that a revoked user may still be able to decrypt an old ciphertext they were authorized to access before being revoked. In order to address this problem, the ciphertext stored in the cloud storage should be updated, ideally by the (untrusted) cloud server. In the literature, proxy re-encryption [3] has been proposed to enable the change of the authorized decryptor of a ciphertext, and this approach has been incorporated into ABE [4]–[7] to allow a ciphertext to be updated by a third party (e.g., the CSP) for revocation purpose. However, the proxy re-encryption approach requires re-encryption/update keys to be issued to the CSP in order to allow the ciphertexts to be updated. From the practicality

Manuscript received September 17, 2017; revised January 6, 2018; accepted February 13, 2018. Date of publication February 27, 2018; date of current version April 16, 2018. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Stefan Katzenbeisser. *(Corresponding author: Guomin Yang.)*

S. Xu, G. Yang, and Y. Mu are with the School of Computing and Information Technology, Institute of Cybersecurity and Cryptology, University of Wollongong, Wollongong, NSW 2522, Australia (e-mail: sx914@uow.edu.au; gyang@uow.edu.au; ymu@uow.edu.au).

R. H. Deng is with the School of Information Systems, Singapore Management University, Singapore 188065 (e-mail: robertdeng@smu.edu.sg).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2018.2810065

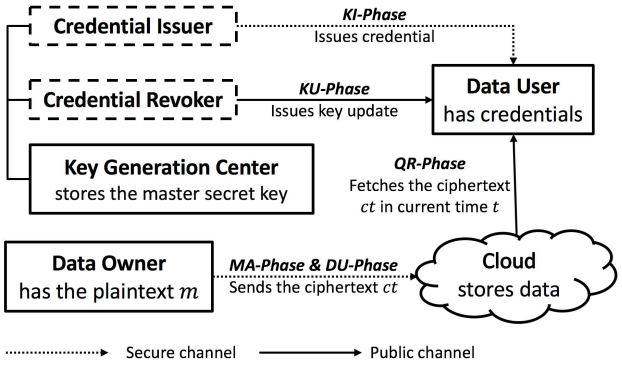

Fig. 1. System Model.

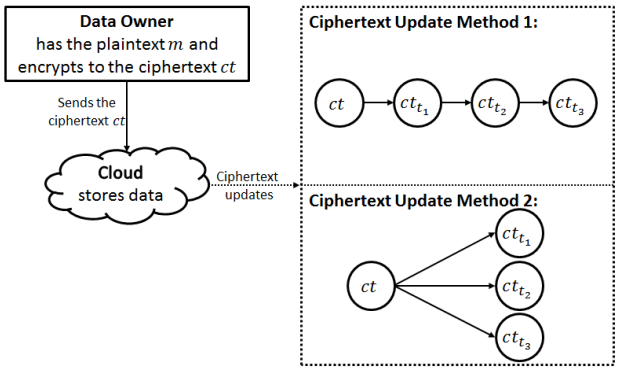

Fig. 2. Ciphertext Update.

perspective, it is more desirable that the ciphertext update, which can happen frequently, can be done by the CSP without requiring any delegated key.

In this work, we aim to develop a secure and practical approach to address the problem. As shown in Fig. 1 and 2, a data owner sends the original ciphertext ct to the cloud storage in revocation epoch t, and a data user can query the cloud storage to access the ciphertext afterwards. Suppose the data user issues a query at $t_1 \geq t$, the CSP then sends the updated ciphertext ct_{t_1} (i.e., ct_{t_1} is transformed from ct by the CSP) to the data user, and this ciphertext can be decrypted by the user if and only if he/she satisfies the access policy specified by the data owner and is not revoked at t_1. Different from the proxy re-encryption based approach, here we'd like the transformation to be performed by the CSP without using any re-encryption/update key.

In Crypto'12, Sahai *et al.* [8] introduced the concept of ciphertext delegation, which allows the ciphertext to be updated by the data storage without accessing any secret information. Following the indirect revocation approach, the private key of each user is divided into an attribute-based secret key and a time-based update key, and the latter is updated in each revocation epoch for non-revoked users. In order to achieve ciphertext delegation, two ABE schemes (or two instances of an ABE) were used by Sahai *et al.'s* RABE scheme to handle the attributes and the time component, respectively.In their concrete scheme, two instances of the dual ABE scheme introduced in [9] were applied. Although Sahai *et al.* [8] introduced the first RABE scheme supporting ciphertext delegation, their scheme is not very practical since two ABE schemes need to be applied in their construction. Also, their concrete RABE scheme with ciphertext delegation based on dual encryption is under composite order groups and hence not efficient. Such a drawback makes their scheme impractical in large-scale applications that involve a large number of users and frequent user membership changes.

A. Our Contribution

In this work, we present a new solution for enabling attribute-based access control for dynamic user groups in cloud storage systems. Specifically, we present a new RABE scheme that allows the cloud storage to update ciphertexts for handling revocation without any delegated key and at the same time achieves high efficiency. The high cost of Sahai *et al.'s* scheme is mainly due to the use of two ABE components for handling the attributes and the time component. Therefore, our idea is to replace the ABE for handling the time component by a more efficient primitive. One difficulty of realizing this idea is that we need a new access control mechanism for the time component to efficiently handle revocation, and the other difficulty is that we need to build a primitive that supports ciphertext update and can be integrated with the new time control mechanism. To address these problems, we first introduce a novel time encoding mechanism and then combine it with a variant of the Waters IBE to achieve our goal. As a result, our construction significantly reduces the computation and storage cost by a factor of $\log \mathcal{T}$ where $\mathcal{T}$ denotes the bounded system life time. An efficiency comparison between Sahai *et al.'s* scheme and ours is presented in section VI.

Based on our RABE scheme, we present a secure and fine-grained access control and data sharing system for cloud-based on-demand service applications. Specifically, we use the cloud-based on-demand movie streaming as a typical example. We show our system provides a practical solution for such applications which demand fine-grained access control and frequent user membership update for large user groups.

We should note that there is a difference between the ciphertext update in Sahai *et al.'s* scheme and in ours. The difference is illustrated in Fig. 2. Suppose a data owner sends the ciphertext ct to the CSPs. In the first method introduced in [8], it allows the CSPs to sequentially update ct to ct_{t_1} and then to ct_{t_3}, and so on. While in our construction (second method), we always let the CSPs to perform the ciphertext update from the original ciphertext ct. Our approach requires the cloud server to only maintain the origianl ciphertext uploaded by the data owner and hence is easier for storage management and maintanance.

B. Related Work

Many cryptographic schemes including IBE only provides a coarse-level access control. It limits the ability of users to selectively share their encrypted data at a fine-grained level. Sahai and Waters [1] made some initial steps to

solve this problem by introducing ABE. To enrich expressiveness of access control policies, Goyal *et al.* [2] and Bethencourt *et al.* [10] proposed tree based key-policy and ciphertext-policy ABE schemes, respectively. In key-policy ABE schemes (KP-ABE), attribute sets are used to annotate ciphertexts, and private keys are associated with access structures that specify which ciphertexts the user will be entitled to decrypt. Ciphertext-policy ABE (CP-ABE) proceeds in a dual way, by assigning attribute sets to private keys and letting senders specify an access policy that receivers' attribute sets should comply with. Due to the sizes of the key and the ciphertext are linearly increased in the universe of the attribute set and security proofs are under the selectively model in [1] and [2], Attrapadung *et al.* [11] proposed the first constant-size ABE and Lewko *et al.* [9] provided first fully secure ABE with dual encryption system [12], respectively.

Same as other cryptographic primitives, efficient user revocation is very important in ABE systems. Pirretti *et al.* [13] suggested extending each attribute with an expiration date, e.g., by representing an attribute as $\mathsf{Att}\|\mathsf{T}$ where Att is the real attribute and T is the expiration date. However, this approach is impractical as the Boneh and Franklin RIBE solution [14]. Bethencourt *et al.* [10] proposed a bit representation to support integer comparisons, and this approach extends the validity period of a key to a couple of revocation epochs.

To prevent revoked users from decrypting old ciphertexts stored in a public cloud storage, a few RABE scheme have been proposed in the literature [4]–[7] to allow ciphertexts to be updated by the cloud server. These RABE constructions incorporated the proxy re-encryption technique where re-encryption/update keys are issued to the cloud server to enable ciphertext update.

Following the first practical RIBE [15] introduced in 2008, a few RABE schemes [8], [16] have been proposed by applying the indirect revocation method. They all follow the idea of dividing the private key into an attribute-based secret key and a time-related update key. The RABE scheme proposed in [8] also supports the concept of revocable storage which means a ciphertext can be updated by a third party without requiring any delegated key. However, the scheme in [8] used ABE to handle both the attributes and the time component, which makes the scheme less efficient.

C. Outline

We introduce some preliminaries in Section II and provide the definition of RABE and its security model in Section III, which is followed by our constructions and their security proofs in Section IV. We then present an application of our RABE and the experimental results in Section V and VI, respectively.

II. Preliminaries

A. Notations

Let $\mathbb{N}$ denote the set of all natural numbers, and for $n \in \mathbb{N}$, we define $[n] := \{1, ..., n\}$. If a and b are strings, then "$|a|$" denotes the bit-length of a, "$a||b$" denotes the concatenation of a and b. Let $\vec{u} := (u_1, u_2, ..., u_\ell)$ denote a vector of dimension ℓ in $\mathbb{Z}_p^\ell$. Let the Greek character "λ" denote a security parameter. A function $\epsilon(\lambda) : \mathbb{N} \to [0, 1]$ is said to be *negligible* if for all positive polynomials $p(\lambda)$ and all sufficiently large $\lambda \in \mathbb{N}$, we have $\epsilon(\lambda) < 1/p(\lambda)$.

B. Bilinear Map

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two cyclic multiplicative groups of prime order p and g be a generator of $\mathbb{G}$. The map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is said to be an admissible bilinear pairing if the following properties hold true.

- Bilinearity: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, $e(u^a, v^b) = e(u, v)^{ab}$.
- Non-degeneration: $e(g, g) \neq 1$.
- Computability: it is efficient to compute $e(u, v)$ for any $u.v \in \mathbb{G}$.

C. The Decisional Bilinear Diffie-Hellman Assumption

Let $a, b, c, z \in \mathbb{Z}_p$ be chosen at random and g be a generator of $\mathbb{G}$. The decisional Bilinear Diffie-Hellman (BDH) assumption [1] is that no probabilistic time algorithm $\mathcal{D}$ can distinguish the tuple $(A = g^a, B = g^b, C = g^c, D = e(g, g)^{abc})$ from the tuple $(A = g^a, B = g^b, C = g^c, Z = e(g, g)^z)$ with a non-negligible advantage over random guess.

D. Access Structure and Monotone Span Program

We recall the definitions of access structures and monotone span program, as defined in [2] and [17].

Definition 1 (Access Structure): Let $\{P_1, ..., P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1,...,P_n\}}$ is monotone if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$, then $C \subseteq \mathbb{A}$. A monotone access structure is a monotone collection $\mathbb{A}$ of non-empty subsets of $\{P_1, ..., P_n\}$, i.e., $\mathbb{A} \subseteq 2^{\{P_1,...,P_n\}} \setminus \{\emptyset\}$. The sets in $\mathbb{A}$ are called authorized sets, and the sets not in $\mathbb{A}$ are called unauthorized sets.

Definition 2 (Monotone Span Program (MSP)): Let $\mathcal{K}$ be a field and $\{x_1, ..., x_n\}$ be a set of variables. A MSP over $\mathcal{K}$ is labeled matrix $\tilde{M}(M, \rho)$ where M is a matrix over $\mathcal{K}$, and ρ is a labeling of the rows of M by literals from $\{x_1, ..., x_n\}$ (every row is labeled by one literal).

A MSP accepts or rejects an input by the following criterion. For every input set S if literals, define the submatrix M_S of M consisting of those rows whose labels are in S, i.e., rows labeled by some i such that $i \in S$. The MSP $\tilde{M}$ accepts S of and only if $\vec{1} \in \mathsf{span}(M_S)$, i.e., some linear combination of the rows of M_S given the all-one vector $\vec{1}$. The MSP $\tilde{M}$ computes a boolean function f_M if it accepts exactly those input S where $f_M(S) = 1$. The size of $\tilde{M}$ is the number of rows in M.

In the rest of paper, we define M as a matrix with $d \times \ell$ elements, where d is a dynamic value depending on the access policy $\mathbb{A}$. M_i stands for the i^{th} row of the matrix M and is a vector size of ℓ. In our proposed scheme, each row of the matrix M maps to different attributes. For simply the notation, let $\mathbb{A}(S) = 1$ indicate the attribute set S satisfies the access policy $\mathbb{A}$ and $\mathbb{A}(S) = 0$ denote the attribute set S does not satisfy the access policy $\mathbb{A}$.

E. Subset-Cover Framework

Boldyreva *et al.* [15] pointed out that the revocation list can be implemented by subset-cover framework (tree based data structure) [18] called Complete Subtree (CS) method. Every user is assigned to an unique leaf node. Let x_l and x_r denote the left child and right child of node x, respectively. The key update nodes algorithm KUNodes takes three parameters as input, a binary tree BT, a revocation list rl and a time t. It outputs a set of nodes, which is the minimal set of nodes in the binary tree BT such that all the non-revoked nodes have at least one ancestor or themselves in the set and none of revoked nodes in revocation list rl has any ancestor or themselves in the set. The function operates as Algorithm 1. First, it marks all the ancestors of revoked nodes as revoked into the set X, then it outputs all the non-revoked children of revoked nodes in the set Y. Let $root$ denote the root node of the binary tree BT. Here is a formal specification.

Algorithm 1 KUNodes Algorithm

```
function KUNodes(BT, rl, t)
    X, Y ← ∅
    for (v_i, t_i) ∈ rl do
        if t_i ≤ t then X ← X ∪ Path(v_i)
        end if
    end for
    for x ∈ X do
        if x_l ∉ X then Y ← Y ∪ x_l
        end if
        if x_r ∉ X then Y ← Y ∪ x_r
        end if
    end for
    if Y = ∅ then Y ← root
    end if
    return Y
end function
```

F. Managing the Time Structure

Bethencourt *et al.* [10] pointed out the key revocation can be realized by numerical bit representation. Inspirited by this, we design a time encoding mechanism TEncode. The details of this algorithm are described in Algorithm 2. It takes the current date t and the bounded system life time $\mathcal{T}$ as input, and outputs a bit string $\tilde{t}$ of the size $\log_2 \mathcal{T}$. The algorithm is quite simply and straightforward. First, we convert a decimal number t to a binary number $\tilde{t}$. Then, we fill zeros as prefix if the size of $\tilde{t}$ is less than $\log_2 \mathcal{T}$.

Algorithm 2 TEncode Algorithm

```
function TEncode(t, T)
    encode the decimal number t to a bit string t̃
    while |t̃| < log_2 T do t̃ ← 0||t̃
    end while
    return t̃
end function
```

We design a ciphertext time encoding algorithm CTEncode to reduce the size of the ciphertext by a factor of $O(\log \mathcal{T})$ without losing the property of public ciphertext update. The details of this algorithm are represented as Algorithm 3. It takes the time t and the bounded system life time $\mathcal{T}$ as input, and outputs a bit string $\tilde{t}$ of the size $\log_2 \mathcal{T}$ that is less than all future revocation epochs. We first run $\mathsf{TEncode}(t, \mathcal{T})$ and receive $\tilde{t}$ as the output. Then, for all i in $[\log_2 \mathcal{T}]$, we modify the bit $\tilde{t}[i]$ as follows:

- If $\tilde{t}[i] = 0$, we set a boolean value chk to true (initial false) that means the first 0-bit is found.
- If $\tilde{t}[i] = 1$ and chk is true, we modify $\tilde{t}[i]$ to 0.
- Otherwise, we do not modify any bit.

Algorithm 3 CTEncode Algorithm

```
function CTEncode(t, T)
    t̃ ← TEncode(t, T)
    chk ← false
    for i ∈ [log_2 T] do
        if t̃[i] = 1 and chk = false then t̃[i] = 1
        else
            chk ← true
            t̃[i] = 0
        end if
    end for
    return t̃
end function
```

The above algorithm is used to reduce the space complexity of the ciphertext and to process ciphertext delegation.

1) Bit Representation: The bit representation reduces the space complexity of the ciphertext. The returning string $\tilde{t}$ is equal or less than the current date t allowing to derive all the time $t' \geq t$. Suppose the bounded system life time is $15 = 1111_2$, we consider the following two cases:

- The most significant bit is non-zero, e.g., the current revocation epoch is $10 = 1010_2$, the output of CTEncode is $8 = 1000_2$.
- The most significant bit is zero. e.g., the current revocation epoch is $5 = 0101_2$, the output of CTEncode is $4 = 0100_2$.

Hence, the output of CTEncode is less than the inputting revocation epoch, and the binary expression is allowed to increase by modifying the 0 to 1.

2) Ciphertext Delegation: We implement the time-related component of the ciphertext in a novel way to achieve ciphertext delegation.

- Producing a set $\mathcal{V}_t$ records all i^{th} bit such that $t[i] = 0$ for the output of CTEncode.
- For each $i \in \mathcal{V}_t$, individually generate the time-related ciphertext component C_i by Waters' IBE.

Hence, the ciphertext component C_i is given only for all i^{th} positions with $t[i] = 0$. The time-related part of the ciphertext at the time $t' \geq t$ can be retrieved by combining the components C_i for all $i \in \mathcal{V}_{t'}$ ($\mathcal{V}_{t'} \subseteq \mathcal{V}_t$ by definition). Note that the updated ciphertext needs to be re-randomized to prevent some trivial attacks.

G. Goyal et al.'s ABE Scheme

For $x, i \in \mathbb{Z}$ and $\mathsf{J} \subset \mathbb{Z}$ the *Lagrange coefficient* $\Delta_{i,\mathsf{J}}(x)$ is defined as

$$\Delta_{i,\mathsf{J}}(x) = \prod_{j \in \mathsf{J}, j \neq i} \left(\frac{x-j}{i-j} \right).$$

Goyal *et al.* [2] provided two concrete constructions. In this section, we review the large universe construction with a monotone span program.

$\mathsf{S}(\lambda, n)$ is the ***setup*** algorithm. It chooses a bilinear group of order p, a random value $\alpha \in \mathbb{Z}_p$, and random generators $g, g_2 \in \mathbb{G}$, then it sets $g_1 = g^\alpha$. For all $i \in [n+1]$ where $n \in \mathbb{N}$ is the maximum size of the attribute set used in encryption, it randomly picks $t_i \in \mathbb{Z}_p$ and defines a function $T(x)$ as:

$$T(x) = g_2^{x^n} \prod_{i=1}^{n+1} t_i^{\Delta_{i,[n+1]}(x)}.$$

It returns

$$pp = (p, g, g_1, g_2, \{t_i\}_{t \in [n+1]}), \quad msk = \alpha.$$

$\mathsf{SK}(msk, (M, \rho))$ is the ***private key generation*** algorithm. It takes msk and an access structure (M, ρ) as input where M is a matrix of the size $d \times \ell$. Let $\vec{u}$ be a random ℓ dimensional vector over $\mathbb{Z}_p$ and $\vec{1} \cdot \vec{u} = \alpha$. For each row i in the matrix M, it randomly chooses $r_i \in \mathbb{Z}_p$ and computes the secret key sk as:

$$K_i^{(0)} = g_2^{M_i \cdot \vec{u}} T(i)^{r_i}, \quad K_i^{(1)} = g^{r_i}.$$

It returns the secret key $sk = \{K_i^{(0)}, K_i^{(1)}\}_{i \in [d]}$.

$\mathsf{E}(pp, S, m)$ is the ***encryption*** algorithm. It takes the public parameter pp, an attribute set S and a message m as input, and outputs the ciphertext ct. It randomly chooses $s \in \mathbb{Z}_p$ and sets:

$$C = m \cdot e(g_1, g_2)^s, \quad C_1 = g^s, \quad \{C_i = T(i)^s\}_{\rho(i) \in S}.$$

It returns the ciphertext $ct = (S, C, C_1, \{C_i\}_{\rho(i) \in S})$.

$\mathsf{D}(pp, sk, ct)$ is the ***decryption*** algorithm.It takes $\sum_{\rho(i) \in S} M_i w_i = \vec{1}$ and recovers the message m by computing:

$$m = C \cdot \prod_{\rho(i) \in S} \left(\frac{e(K_i^{(1)}, C_i)}{e(K_i^{(0)}, C_1)} \right)^{w_i}$$

Goyal *et al.'s* ABE is secure if the decisional BDH assumption holds.

Theorem 1: The Goyal et al.'s ABE scheme is IND-ABE-CPA secure under decisional BDH assumption.

H. Variant of Waters' IBE Scheme

Waters [19] proposed a practical IBE proven secure under decisional BDH assumption. We slightly modify the scheme as follows.

$\mathsf{S}(\lambda, n)$ is the ***setup*** algorithm. It chooses a bilinear group of order p, picks random generators $g, g_2 \in \mathbb{G}$, and a random integer $\alpha \in \mathbb{Z}_p$. Then it sets $g_1 = g^\alpha$. Additionally, it chooses $U' \in \mathbb{G}$ and for each $i \in [n]$ where n is the size of the user identities in binary form, randomly picks $U_i \in \mathbb{G}$, then it returns the public parameter pp and the master secret key msk as:

$$pp = (p, g, g_1, g_2, U', \{U_i\}_{i \in [n]}), \quad msk = \alpha.$$

$\mathsf{SK}(msk, id)$ is the ***private key generation*** algorithm. Let $\mathcal{V}$ be the set of i where $id[i] = 0$. It randomly chooses $r \in \mathbb{Z}_p$ and computes the secret key sk_{id} as

$$K^{(0)} = g_2^\alpha \left(U' \prod_{i \in \mathcal{V}} U_i \right)^r, \quad K^{(1)} = g^r.$$

Note that Waters' IBE defines $\mathcal{V}$ as the set of i with $id[i] = 1$, and we modify this part to all i with $id[i] = 0$.

$\mathsf{E}(pp, id, m)$ is the ***encryption*** algorithm. It randomly chooses $s \in \mathbb{Z}_p$ and constructs the ciphertext ct as follows:

$$C = m \cdot e(g_1, g_2)^s, \quad C_1 = g^s, \; C_2 = \left(U' \prod_{i \in \mathcal{V}} U_i \right)^s.$$

It returns the ciphertext $ct = (C, C_1, C_2)$.

$\mathsf{D}(pp, sk_{id}, ct)$ is the ***decryption*** algorithm. It outputs the message m by computing

$$m = C \cdot \frac{e(K^{(1)}, C_2)}{e(K^{(0)}, C_1)}$$

The security proof of modified IBE can be derived from Waters' IBE [19]. Our scheme is essentially identical to Waters' IBE except that we define $\mathcal{V}$ to be the set of all i with $id[i] = 0$ rather than $id[i] = 1$ and defines the master secret key as α directly rather than g_2^α for unifying the keys in both ABE and IBE.

Theorem 2: The above variant of Waters' IBE is IND-ID-CPA secure under decisional BDH assumption.

III. RABE and Its Security

In this section, we provide the formal definition and the security model of our RABE scheme. In the rest of paper, let $\mathcal{T}, \mathcal{M}, \mathcal{I}$ be the system life time, the message space, and the identity space, respectively.

A. Syntax of RABE

Definition 3 (RABE): An attribute-based encryption with efficient user revocation and ciphertext delegation (or simply RABE) scheme $\mathsf{RABE} = (\mathsf{S}, \mathsf{SK}, \mathsf{KU}, \mathsf{DK}, \mathsf{E}, \mathsf{CU}, \mathsf{D}, \mathsf{R})$ *is defined by eight algorithms. Each algorithm is run by one of four types of parties - KGC, sender, receiver and CSP. The KGC maintains a revocation list rl and a state st. In what follows, we call an algorithm stateful if it updates the revocation list rl or the state st.*

$\mathsf{S}(\lambda, \mathcal{N}, \mathcal{T}, n)$ *is a stateful* ***setup*** *algorithm run by the KGC. It takes the security parameter λ, the number of users $\mathcal{N}$, the bounded system life time $\mathcal{T}$ and a value n as the maximum number of attributes to be used in encryption as input, and outputs the public parameter pp, the master secret key msk, the revocation list rl (initially empty) and the state st.*

$\mathsf{SK}(msk, st, id, \mathbb{A})$ *is a stateful* ***private key generation*** *algorithm run by the KGC. It takes the master secret key msk, the state st, an identity* $id \in \mathcal{I}$ *and an access structure* $\mathbb{A}$ *as input, and outputs the secret key* sk_{id} *and an updated state st.*

$\mathsf{KU}(msk, rl, st, t)$ *is a probabilistic* ***key update*** *algorithm run by the KGC. It takes the master secret key msk, the revocation list rl, the state st and the revocation epoch* $t \in \mathcal{T}$ *as input, and outputs the key update* ku_t.

$\mathsf{DK}(sk_{id}, ku_t)$ *is a deterministic* ***decryption key generation*** *algorithm run by the receiver. It takes the secret key* sk_{id} *and the key update information* ku_t *as input, and outputs the decryption key* $dk_{id,t}$ *or a special symbol* $\perp$ *indicating that id was revoked.*

$\mathsf{E}(pp, S, t, m)$ *is a probabilistic* ***encryption*** *algorithm run by the sender. It takes the public parameter pp, an attribute set S, an encryption time* $t \in \mathcal{T}$ *and a message* $m \in \mathcal{M}$ *as input, and outputs the original ciphertext ct.*

$\mathsf{CU}(pp, ct, t)$ *is a probabilistic* ***ciphertext update*** *algorithm run by the CSP. It takes the public parameter pp, the ciphertext ct and the time* $t \in \mathcal{T}$ *as input, and outputs an updated ciphertext* ct_t *or a special symbol* $\perp$ *indicating that ciphertext ct is invalid.*

$\mathsf{D}(pp, dk_{id,t}, ct_t)$ *is a deterministic* ***decryption*** *algorithm run by the receiver. It takes the public parameter pp, the decryption key* $dk_{id,t}$ *and the ciphertext* ct_t *as input, and outputs the message* $m \in \mathcal{M}$ *or a special symbol* $\perp$ *indicating that the ciphertext* ct_t *is invalid.*

$\mathsf{R}(rl, id, t)$ *is a stateful* ***revocation*** *algorithm run by the KGC. It takes the revocation list rl, an identity to be revoked* $id \in \mathcal{I}$ *and the revocation time* $t \in \mathcal{T}$ *as input, and outputs an updated revocation list rl.*

B. Security of RABE

Below we describe the security definition of indistinguishability under chosen plaintext attack (IND-RABE-CPA security) between an adversary and a challenger.

Setup: The challenger runs $\mathsf{S}(\lambda, \mathcal{N}, \mathcal{T}, n)$ to setup the RABE system. Then, it gives the public parameter pp to the adversary, and keeps the master secret key msk, an initially empty revocation list rl and a state st.

Phase 1: The adversary is given oracles, including *private key generation* oracle $\mathcal{O}_{\mathsf{SK}}(\cdot,\cdot)$, *key update* oracle $\mathcal{O}_{\mathsf{KU}}(\cdot)$, *revocation* oracle $\mathcal{O}_{\mathsf{R}}(\cdot,\cdot)$, until it signals the phase is over. Following is the definition of the above oracles.

The *private key generation* oracle $\mathcal{O}_{\mathsf{SK}}(\cdot,\cdot)$ takes an identity id and an access structure $\mathbb{A}$ as input, and runs ***private key generation*** algorithm $\mathsf{SK}(msk, st, id, \mathbb{A})$ to return the private key sk_{id}.

The *key update* oracle $\mathcal{O}_{\mathsf{KU}}(\cdot)$ takes the time t as input, and returns key update ku_t generated from ***key update*** algorithm $\mathsf{KU}(msk, rl, st, t)$.

The *revocation* oracle $\mathcal{O}_{\mathsf{R}}(\cdot,\cdot)$ takes the time t and a revoked identity id as input, and runs ***revocation*** algorithm $\mathsf{R}(rl, id, t)$ to return an updated revocation list rl.

Challenge: The adversary outputs two messages m_0 and m_1 of the same size, an attribute set S^* and a time period $t^* \in \mathcal{T}$ satisfying the following constraints:

1) For any *private key generation* query $\mathcal{O}_{\mathsf{SK}}(id, \mathbb{A})$ such that the challenge attribute set S^* satisfies the access structure $\mathbb{A}$, namely $\mathbb{A}(S^*) = 1$, the $\mathcal{O}_{\mathsf{R}}(id, t)$ must be queried on $t \leq t^*$.
2) If a non-revoked user with the identity id whose access structure $\mathbb{A}$ satisfies the challenge attribute set S^*, then id should not be previously queried to *private key generation* oracle.

The challenger randomly picks a bit $b \in \{0, 1\}$, and forwards the challenge ciphertext ct^* to the adversary by running ***encryption*** algorithm $\mathsf{E}(pp, S^*, t^*, m_b)$.

Phase 2: The adversary continues issuing queries to the challenger as in Phase 1, and following the restriction defined in the challenge phase.

Guess: The adversary makes a guess b' for b, and it wins the game if $b = b'$.

Definition 4: An RABE scheme is secure if for any probabilistic polynomial time adversary $\mathcal{A}$, the advantage of this adversary in the above RABE game defined as

$$\Pr[b = b'] - 1/2$$

is negligible in the security parameter λ.

In this paper, we will focus on the selective security which can be defined by letting the adversary commit the challenge attribute set S^* at the beginning of the adversarial game. Note that we do not require the adversary to commit the challenge time t^* as we will use the adaptively secure Waters IBE to handle the time component.

IV. Our RABE Constructions

In this section, we present two constructions of RABE and analyze their security. Our first scheme is RABE with large attribute universe in the standard model. To improve efficiency, we also present the second construction utilizing two cryptographic hash functions modeled as random oracles.

A. Ideas Behind Our Constructions

Our schemes have the master secret key $\alpha \in \mathbb{Z}_p$ and the message hiding component $e(g_1, g_2)^s$, where $g_1 = g^\alpha$, g_2 is a random element in $\mathbb{G}$ and s is a random value in $\mathbb{Z}_p$. The data user is able to decrypt the ciphertext if he/she has the capability to retrieve the message hiding component $e(g_1, g_2)^s$. Our idea is to separate this hiding component into two parts: one is related to the attribute key, and the other one is related to the time-related key.

To recover the attribute key, let $\mathcal{N}$ denote the maximum number of users in the system. Each data user is assigned to a leaf of the binary tree BT with ℓ depth. In the private key generation phase, the KGC assigns each user to an unassigned leaf node and stores his/her identity in this node. Then, the KGC generates private keys based on Goyal *et al.'s* ABE under the secret key α_x, where α_x is the secret information in the node x. Hence, this private key only has partial secret key information α_x that allows the data user to generate the partial message hiding component $e(g, g_2)^{s\alpha_x}$. Thus, the whole message hiding component cannot be recovered.

To recover the rest of message hiding component, let $\mathcal{T}$ denote the bounded system life time. This bounded time will be treated as the identity space in Waters' IBE. In the key update phase, the KGC generates the update information based on Waters' IBE under the secret key $\alpha-\alpha_x$ for the nodes output by $\mathsf{KUNodes}$, which is the minimum set of nodes needs to be updated. The non-revoked data users will obtain the key update information to recover $e(g,g_2)^{s(\alpha-\alpha_x)}$.

Due to the property of key homomorphism, a non-revoked data user recovers the message hiding component by combing attribute-related and time-related components as follows:

$$e(g,g_2)^{s\alpha_x}\cdot e(g,g_2)^{s(\alpha-\alpha_x)}=e(g,g_2)^{s\alpha}=e(g_1,g_2)^s.$$

To achieve the ciphertext delegation, we use the algorithm $\mathsf{CTEncode}$ to handle timestamp. It takes the current revocation epoch t and the bounded system life time $\mathcal{T}$ as input, and outputs a bit string $\tilde{t}$. For more details please refer to Section II-F.

B. Our RABE Scheme in the Standard Model

$\mathsf{S}(\lambda,\mathcal{N},\mathcal{T},n)$ is a probabilistic ***setup*** algorithm. It chooses a bilinear group of order p according to the bilinear group parameter generator $\mathcal{G}(\lambda)$. Then, it selects a random value $\alpha\in\mathbb{Z}_p$, random generators $g,g_2\in\mathbb{G}$, and sets $g_1=g^\alpha$. It picks a binary tree BT with at least $\mathcal{N}$ leaves and generates the relative information about n and $\mathcal{T}$, respectively.

- For each $i\in[n+1]$, it randomly picks $t_i\in\mathbb{Z}_p$ and defines a function $T(x)$ as in Section II-G.
- It chooses $U'\in\mathbb{G}$ and for each $j\in[\log_2\mathcal{T}]$, randomly picks $U_j\in\mathbb{G}$.

Finally, it returns the revocation list $rl=\emptyset$, the state $st=\mathsf{BT}$, the public key pp and the master secret key msk:

$$pp=(p,g,g_1,g_2,\{t_i\}_{i\in[n+1]},U',\{U_j\}_{j\in[\log_2\mathcal{T}]}),msk=\alpha.$$

$\mathsf{SK}(msk,st,id,(M,\rho))$ is a probabilistic ***private key generation*** algorithm. The parameter (M,ρ) defines the access policy for attributes, where M is a matrix of the size $d\times\ell$. It firstly chooses an unassigned leaf node θ from the binary tree BT, and stores id in this node. Then for each node $x\in\mathsf{Path}(\theta)$, it runs as follows:

- It fetches α_x from the node x. If α_x has not been defined, it randomly chooses $\alpha_x\in\mathbb{Z}_p$ and stores it in the node x.
- Let $\vec{u}$ be a random ℓ dimensional vector over $\mathbb{Z}_p$ such that $\vec{1}\cdot\vec{u}=\alpha_x$. For each row i in the matrix M, it randomly chooses $r_i\in\mathbb{Z}_p$ and sets the secret key

$$sk_{id,x}=\{K_i^{(0)}=g_2^{M_i\cdot\vec{u}}T(i)^{r_i},K_i^{(1)}=g^{r_i}\}_{i\in[d]}.$$

Finally, it returns the secret key $sk_{id}=\{sk_{id,x}\}_{x\in\mathsf{Path}(\theta)}$ and an updated state $st=\mathsf{BT}$.

$\mathsf{KU}(msk,rl,st,t)$ is a probabilistic ***key update*** algorithm. It encodes time by running $\tilde{t}\leftarrow\mathsf{TEncode}(t,\mathcal{T})$. Let $\mathcal{V}\in[\log_2\mathcal{T}]$ be the set of all i for which $\tilde{t}[i]=0$. Then, for each node $x\in\mathsf{KUNodes}(\mathsf{BT},rl,t)$, the key update is constructed as:

- It fetches α_x (α_x is always predefined in the **private key generation** algorithm).
- It randomly chooses $r\in\mathbb{Z}_p$, and outputs the key update information $ku_{t,x}=(K^{(0)},K^{(1)})$ as:

$$K^{(0)}=g_2^{\alpha-\alpha_x}\left(U'\prod_{i\in\mathcal{V}}U_i\right)^r,\quad K^{(1)}=g^r.$$

Finally, it returns $ku_t=\{ku_{t,x}\}_{x\in\mathsf{KUNodes}(\mathsf{BT},rl,t)}$.

$\mathsf{DK}(sk_{id},ku_t)$ is a deterministic ***decryption key generation*** algorithm. Parse the secret key sk_{id} as $\{sk_{id,x}\}_{x\in\mathsf{Path}(\theta)}$ and the key update ku_t as $\{ku_{t,x}\}_{x\in\mathsf{KUNodes}(\mathsf{BT},rl,t)}$. Denote I as $\mathsf{Path}(\theta)$, and J as $\mathsf{KUNodes}(\mathsf{BT},rl,t)$. If $\mathsf{I}\cap\mathsf{J}=\emptyset$, it returns $\perp$. Otherwise, it outputs the decryption key $dk_{id,t}=\{sk_{id,x},ku_{t,x}\}_{x\in\mathsf{I}\cap\mathsf{J}}$.

$\mathsf{E}(pp,S,t,m)\rightarrow ct$ is a probabilistic ***encryption*** algorithm. It encodes the time $t\in\mathcal{T}$ to $\tilde{t}\leftarrow\mathsf{CTEncode}(t,\mathcal{T})$. Then, it randomly chooses $s\in\mathbb{Z}_p$ and generates the ciphertext ct:

$$C=me(g_1,g_2)^s,C_1=g^s,\{C_i^{(0)}=T(i)^s\}_{\rho(i)\in S},$$
$$C^{(1)}=U'^s,\{C_j^{(1)}=U_i^s\}_{j\in\mathcal{V}}.$$

It then returns the ciphertext

$$ct=(S,t,C,C_1,\{C_i^{(0)}\}_{\rho(i)\in S},C^{(1)},\{C_j^{(1)}\}_{j\in\mathcal{V}}).$$

$\mathsf{CU}(pp,ct,t)$ is a probabilistic ***ciphertext update*** algorithm. Parse the original ciphertext ct as

$$ct=(S,t',C',C_1',\{C_i'^{(0)}\}_{\rho(i)\in S},C'^{(1)},\{C_j'^{(1)}\}_{j\in\mathcal{V}}).$$

If the time t' in ciphertext is greater than t, it returns $\perp$ indicating the time t is invalid. Otherwise, it encodes $t\in\mathcal{T}$ to a bit string $\tilde{t}\leftarrow\mathsf{TEncode}(t,\mathcal{T})$, randomly chooses $s\in\mathbb{Z}_p$, and re-randomizes the ciphertext ct to ct_t as follows.

$$C=C'e(g_1,g_2)^s=me(g_1,g_2)^{s+s'},$$
$$C_1=C_1'g^s=g^{s+s'},C_i=C_i'^{(0)}T(i)^s=T(i)^{s+s'},$$
$$C_t=C'^{(1)}U'^s\prod_{j\in\mathcal{V}}C_j'^{(1)}U_j^s=\left(U'\prod_{j\in\mathcal{V}}U_j\right)^{s+s'}.$$

It then returns the updated ciphertext ct_t:

$$ct_t=(S,t,C,C_1,\{C_i\}_{\rho(i)\in S},C_t).$$

$\mathsf{D}(pp,dk_{id,t},ct_t)$ is a deterministic ***decryption*** algorithm. Parse the decryption key $dk_{id,t}=\{sk_{id,x},ku_{t,x}\}_{x\in\mathsf{I}\cap\mathsf{J}}$. The message m can be covered from ct_t as follows:

- Parse $sk_{id,x}=\{K_i^{(0)},K_i^{(1)}\}_{i\in[d]}$. Recover the secret information in the attribute-related component by taking $\sum_{\rho(i)\in S}M_iw_i=\vec{1}$ and computing A_1:

$$\begin{aligned}A_1&=\prod_{\rho(i)\in S}\left(\frac{e(K_i^{(0)},C_1)}{e(K_i^{(1)},C_i^{(0)})}\right)^{w_i}\\&=\prod_{\rho(i)\in S}\left(\frac{e(g_2^{M_i\cdot\vec{u}},g^s)e(T(i)^{r_i},g^s)}{e(g^{r_i},T(i)^s)}\right)^{w_i}\\&=\prod_{\rho(i)\in S}e(g_2^{M_i\cdot\vec{u}},g^s)^{w_i}\\&=e(g,g_2)^{s\alpha_x}.\end{aligned}$$

- Parse $ku_{t,x} = (K^{(0)}, K^{(1)})$. Recover the secret information in the time-related component by computing A_2:

$$\begin{aligned} A_2 &= \frac{e(K^{(0)}, C_1)}{e(K^{(1)}, C_t)} \\ &= \frac{e(g_2^{\alpha-\alpha_x}, g^s)e\left(\left(U'\prod_{i\in\mathcal{V}} U_i\right)^r, g^s\right)}{e\left(g^r, \left(U'\prod_{j\in\mathcal{V}} U_j\right)^s\right)} \\ &= e(g, g_2)^{s(\alpha-\alpha_x)}. \end{aligned}$$

- Recover the message blinding component A_3:

$$A_3 = A_1 A_2 = e(g, g_2)^{s\alpha} = e(g_1, g_2)^s.$$

Finally, it recovers and returns the message m as:

$$C/A_3 = m \cdot e(g_1, g_2)^s / e(g_1, g_2)^s = m.$$

$\mathsf{R}(rl, id, t)$ is a deterministic **revocation** algorithm. It adds (id, t) to rl, and returns the updated revocation list rl.

C. Security Analysis of Our RABE in the Standard Model

We prove the security of our scheme in the selective security model by reducing the security of Goyal *et al.'s* ABE [2] and Waters IBE [19] to the security of our proposed scheme. Since we handle the attributes and access policies by following Goyal *et al.'s* ABE scheme, our RABE construction is also secure against collusions among users.

Theorem 3: Our first RABE scheme is selectively IND-RABE-CPA secure in the standard model under the assumption that Goyal et al.' ABE scheme is selective IND-ABE-CPA secure and Waters' IBE scheme is IND-ID-CPA secure.

Proof: Suppose there exists a probabilistic polynomial time adversary $\mathcal{A}$ that can break our RABE scheme in the selective security model with a non-negligible advantage ϵ. We build an algorithm $\mathcal{B}$ that can have a non-negligible advantage in the ABE security game simulated by $\mathcal{C}_{\mathsf{ABE}}$ or the IBE security game simulated by $\mathcal{C}_{\mathsf{IBE}}$. Before describing $\mathcal{B}$, we define two types of the adversary.

- $\mathcal{A}$ is a non-revoked user who cannot query the private key of the challenge attribute set S^* but is allowed to gain the key update information in t^*. In this case, $\mathcal{B}$ can break the game of $\mathcal{C}_{\mathsf{ABE}}$.
- $\mathcal{A}$ is a revoked user who can query the private key of the challenge attribute set S^* and the key update information in t^* but all users with $\mathbb{A}(S^*) = 1$ must be revoked before or at the challenge time t^*. In this case, $\mathcal{B}$ can break the game of $\mathcal{C}_{\mathsf{IBE}}$.

The reduction works differently for each type of adversary. $\mathcal{B}$ proceeds as follows:

Init: $\mathcal{B}$ runs $\mathcal{A}$. $\mathcal{A}$ chooses the challenge attribute set S^* and sends S^* to $\mathcal{B}$.

Setup: $\mathcal{B}$ picks a leaf node θ^* and randomly chooses a bit $rev \in \{0, 1\}$.

If $rev = 0$, $\mathcal{B}$ assumes $\mathcal{A}$ is a non-revoked user and chooses $\mathcal{C}_{\mathsf{ABE}}$ to break. $\mathcal{B}$ sends S^* to $\mathcal{C}_{\mathsf{ABE}}$. $\mathcal{C}_{\mathsf{ABE}}$ returns the message $(p, g, g_1, g_2, \{t_i\}_{i\in[n+1]})$. $\mathcal{B}$ then randomly chooses $r_{u'} \in \mathbb{Z}_p$ and computes $U' = g^{r_{u'}}$. For all $i \in [\log_2 \mathcal{T}]$, $\mathcal{B}$ randomly picks $r_{u_i} \in \mathbb{Z}_p$ and sets $U_i = g^{r_{u_i}}$.

If $rev = 1$, $\mathcal{B}$ assumes $\mathcal{A}$ is a revoked user and chooses $\mathcal{C}_{\mathsf{IBE}}$ to break. $\mathcal{B}$ receives the public information $(p, g, g_1, g_2, U', \{U_j\}_{j\in[\log_2 \mathcal{T}]})$ from $\mathcal{C}_{\mathsf{IBE}}$, and chooses two random n degree polynomials $f(x)$ and $u(x)$, where $u(x) = -x^n$ for $\rho(x) \in S^*$ and $u(x) \neq -x^n$ otherwise. Then $\mathcal{B}$ sets $t_i = g_2^{u(i)} g^{f(i)}$ for all $i \in [n+1]$, which implicitly defines $T(i) = g^{f(i)}$ for $\rho(x) \in S^*$ and $T(i) = g_2^{i^n+u(i)} g^{f(i)}$ otherwise.

$\mathcal{B}$ sends the public parameter pp to $\mathcal{A}$:

$$pp = (p, g, g_1, g_2, \{t_i\}_{t\in[n+1]}, U', \{U_j\}_{j\in[\log_2 \mathcal{T}]}).$$

$\mathcal{B}$ sets $rl = \emptyset$ and $st = \mathsf{BT}$, where BT is a binary tree with at least $\mathcal{N}$ leaves.

Phase 1: $\mathcal{A}$ adaptively queries the following oracles.

$\mathcal{O}_{\mathsf{SK}}(id, \mathbb{A})$: $\mathcal{A}$ makes the private key generation query with the identity id and an access structure $\mathbb{A}$, where $\mathbb{A}$ is (M, ρ) and M is a matrix with the size of $d \times \ell$. $\mathcal{B}$ answers the query $(id, \mathbb{A})$ as follows:

Case 1: If $rev = 0$ and $\mathbb{A}(S^*) = 1$, it aborts since $\mathcal{A}$ is a non-revoked user and the private key for $\mathbb{A}(S^*) = 1$ cannot be queried.

Case 2: If $rev = 0$ and $\mathbb{A}(S^*) = 0$, $\mathcal{B}$ sends the access structure $\mathbb{A}$ to $\mathcal{C}_{\mathsf{ABE}}$. $\mathcal{B}$ then receives $\{(K_i^{(0)}, K_i^{(1)})\}_{i\in[d]}$ as the secret key for the access structure $\mathbb{A}$. According to [2, Proposition 1], we have

$$M_i \vec{u} = \vec{v} + \frac{\alpha - \vec{v}}{h} \cdot \vec{w} = \alpha\mu_1 + \mu_2,$$

where the coefficients $\mu_1 = M_i \vec{w} \cdot h^{-1}$ and $\mu_2 = M_i(h\vec{v} - \vec{v}\vec{w})$ are computable.

To simulate the private key for id, $\mathcal{B}$ picks an unassigned leaf node θ from BT and stores id in node θ. For all nodes $x \in \mathsf{Path}(\theta)$, it fetches α_x. If α_x has not been defined, it then randomly chooses $\alpha_x \in \mathbb{Z}_p$ and stores α_x in node x. To generate the private key hiding the secret key $\alpha - \alpha_x$, we have

$$M_i \vec{u} = \vec{v} + \frac{\alpha - \alpha_x - \vec{v}}{h} \cdot \vec{w} = \alpha\mu_1 + \mu_2 + \mu_3,$$

where $\mu_3 = -M_i \vec{w} \alpha_x$ is computable. $\mathcal{B}$ randomly chooses $r_i \in \mathbb{Z}_p$ and generates the secret key $sk_{id,x}$ as

$$sk_{id,x} = \{(K_i^{(0)} g_2^{\mu_3} T(i)^{r_i}, K_i^{(1)} g^{r_i})\}_{i\in[d]}.$$

$\mathcal{B}$ returns the secret key $sk_{id} = \{sk_{id,x}\}_{x\in\mathsf{Path}(\theta)}$

Case 3: $rev = 1$ and $\mathbb{A}(S^*) = 0$. To generate the private key hiding the secret α, $\mathcal{B}$ randomly chooses $r'_x \in \mathbb{Z}_p$ and sets $\{K_i^{(0)}, K_i^{(1)}\}_{i\in[d]}$ as

$$K_i^{(0)} = g_1^{\frac{-\mu_1 \cdot f(i)}{i^n + u(i)}} g_2^{\mu_2} T(i)^{r'_x}, \qquad K_i^{(1)} = g^{r'_x} g_1^{\frac{-\mu_1}{i^n+u(i)}},$$

where $\{K_i^{(0)}, K_i^{(1)}\}_{i\in[d]}$ is legitimate key since $i^n + u(i) \neq 0$ since $\mathbb{A}(S^*) = 0$ and we have

$$M_i \vec{u} = b\vec{v} + \frac{ab - b\vec{v}}{h} \cdot \vec{w} = ab\mu_1 + b\mu_2 = b(a\mu_1 + \mu_2).$$

Thus, for each row i in M, the valid secret key has to contain the component

$$g^{b(a\mu_1+\mu_2)} = g_2^{a\mu_1+\mu_2}.$$

If we set $r_x = r'_x - \frac{a\mu_1}{i^n+u(i)}$, then

$$\begin{aligned}
K_i^{(0)} &= g_1^{\frac{-\mu_1 \cdot f(i)}{i^n+u(i)}} g_2^{\mu_2} T(i)^{r'_x} \\
&= g_1^{\frac{-\mu_1 \cdot f(i)}{i^n+u(i)}} g_2^{\mu_2} (g_2^{i^n+u(i)} g^{f(i)})^{r'_x} \\
&= g_2^{\mu_2} g^{\frac{-a\mu_1 \cdot f(i)}{i^n+u(i)}} (g_2^{i^n+u(i)} g^{f(i)})^{r'_x} \\
&= g_2^{a\mu_1} g_2^{\mu_2} (g_2^{-a\mu_1} g^{\frac{-a\mu_1 \cdot f(i)}{i^n+u(i)}}) (g_2^{i^n+u(i)} g^{f(i)})^{r'_x} \\
&= g_2^{a\mu_1} g_2^{\mu_2} (g_2^{i^n+u(i)} g^{f(i)})^{\frac{-a\mu_1}{i^n+u(i)}} (g_2^{i^n+u(i)} g^{f(i)})^{r'_x} \\
&= g_2^{a\mu_1} g_2^{\mu_2} (g_2^{i^n+u(i)} g^{f(i)})^{r'_x - \frac{a\mu_1}{i^n+u(i)}} \\
&= g_2^{a\mu_1+\mu_2} T(x)^{r'_x - \frac{a\mu_1}{i^n+u(i)}} \\
&= g_2^{a\mu_1+\mu_2} T(x)^{r_x}
\end{aligned}$$

and,

$$K_i^{(1)} = g^{r'_x} g_1^{\frac{-\mu_1}{i^n+u(i)}} = g^{r'_x - \frac{a\mu_1}{i^n+u(i)}} = g^{r_x}$$

Therefore, $\mathcal{B}$ can construct the private key for the access structure $\mathbb{A}$.

$\mathcal{B}$ picks an unassigned leaf node θ from BT, stores id in this node θ and simulates private keys for all nodes $x \in \mathsf{Path}(\theta)$. It firstly fetches α_x. If α_x has not been defined, it then randomly chooses $\alpha_x \in \mathbb{Z}_p$ and stores α_x in the node x.

Case 3.1: $x \in (\mathsf{Path}(\theta) \setminus \mathsf{Path}(\theta^*))$. Parse $\mu_3 = -M_i \vec{w} \alpha_x$. We have

$$\begin{aligned}
M_i \vec{u} &= b\vec{v} + \frac{ab - \alpha_x - b\vec{v}}{h} \cdot \vec{w} \\
&= ab\mu_1 + b\mu_2 + m u_3 \\
&= b(a\mu_1 + \mu_2) + \mu_3.
\end{aligned}$$

$\mathcal{B}$ chooses $r_i \in \mathbb{Z}_p$ and generates $sk_{id,x}$ as follows:

$$sk_{id,x} = \{(K_i^{(0)} g^{\mu_3} T(i)^{r_i}, K_i^{(1)} g^{r_i})\}_{i \in [d]}.$$

Case 3.2: $x \in (\mathsf{Path}(\theta) \cap \mathsf{Path}(\theta^*))$. Taking $\vec{1} \cdot \vec{u} = \alpha_x$. For each row i in the matrix M, it randomly chooses $r_i \in \mathbb{Z}_p$ and computes secret key $sk_{id,x}$ as:

$$sk_{id,x} = \{(g_2^{M_i \cdot \vec{u}} T(i)^{r_i}, g^{r_i})\}_{i \in [d]}.$$

$\mathcal{B}$ returns $sk_{id} = \{sk_{id,x}\}_{x \in \mathsf{Path}(\theta)}$ to $\mathcal{A}$.

Case 4: $rev = 1$ and $\mathbb{A}(S^*) = 1$. $\mathcal{B}$ assigns the node θ^* to id^*. For all nodes $x \in \mathsf{Path}(\theta^*)$, it fetches α_x and if α_x has not been defined, it then randomly chooses $\alpha_x \in \mathbb{Z}_p^*$ and stores α_x in the node x. For each row i in matrix M, it randomly chooses $r_i \in \mathbb{Z}_p$ and computes secret key $sk_{id,x} = \{(g_2^{M_i \cdot \vec{u}} T(i)^{r_i}, g^{r_i})\}_{i \in [d]}$ by taking $\vec{1} \cdot \vec{u} = \alpha_x$.

$\mathcal{B}$ returns $sk_{id} = \{sk_{id,x}\}_{x \in \mathsf{Path}(\theta^*)}$ to $\mathcal{A}$.

$\mathcal{O}_{\mathsf{KU}}(t)$: $\mathcal{A}$ makes the key update query with the time t.

Case 1: $rev = 0$. For each $x \in \mathsf{KUNodes}(\mathsf{BT}, rl, t)$, it fetches α_x. It then picks a random value $r \in \mathbb{Z}_p$ and computes the key update information $ku_{t,x}$ as:

$$Ku_{t,x} = \left(g_2^{\alpha_x} \left(U' \prod_{i \in \mathcal{V}} U_i \right)^r, g^r \right).$$

$\mathcal{B}$ returns $ku_t = \{ku_{t,x}\}_{x \in \mathsf{KUNodes}(\mathsf{BT}, rl, t)}$ to $\mathcal{A}$.

Case 2: $rev = 1$ and $\mathbb{A}(S^*) = 0$. For each $x \in \mathsf{KUNodes}(\mathsf{BT}, rl, t)$, it fetches α_x. Then,

Case 2.1: $x \in (\mathsf{Path}(\theta) \setminus \mathsf{Path}(\theta^*))$. $\mathcal{B}$ picks a random value $r \in \mathbb{Z}_p$ and generates $ku_{t,x}$ as

$$ku_{t,x} = \left(g_2^{\alpha_x} \left(U' \prod_{i \in \mathcal{V}} U_i \right)^r, g^r \right).$$

Case 2.2: $x \in (\mathsf{Path}(\theta) \cap \mathsf{Path}(\theta^*))$. $\mathcal{B}$ forwards the message t to $\mathcal{C}_{\mathsf{IBE}}$, and $\mathcal{C}_{\mathsf{IBE}}$ returns $(K^{(0)}, K^{(1)})$. $\mathcal{B}$ picks a random value $r \in \mathbb{Z}_p$ and generates $ku_{t,x}$ as

$$ku_{t,x} = \left(K^{(0)} g_2^{-\alpha_x} \left(U' \prod_{i \in \mathcal{V}} U_i \right)^r, K^{(1)} g^r \right).$$

Case 3: $rev = 1$ and $\mathbb{A}(S^*) = 1$. For each $x \in \mathsf{KUNodes}(\mathsf{BT}, rl, t)$, it fetches α_x. Then, $\mathcal{B}$ forwards the message t to $\mathcal{C}_{\mathsf{IBE}}$, and $\mathcal{C}_{\mathsf{IBE}}$ returns $(K^{(0)}, K^{(1)})$. $\mathcal{B}$ picks a random value $r \in \mathbb{Z}_p$ and generates $ku_{t,x}$ as

$$ku_{t,x} = \left(K^{(0)} g_2^{-\alpha_x} \left(U' \prod_{i \in \mathcal{V}} U_i \right)^r, K^{(1)} g^r \right).$$

$\mathcal{O}_{\mathsf{R}}(id, t)$: $\mathcal{A}$ makes the revocation query with the identity id and the time t. It sets the revocation list $rl \leftarrow rl \cup (id, t)$.

Challenge: $\mathcal{A}$ will submit two messages (m_0, m_1) and the challenge time t^* to $\mathcal{B}$.

Case 1: $rev = 0$. $\mathcal{B}$ forwards m_0 and m_1 to $\mathcal{C}_{\mathsf{ABE}}$ and $\mathcal{C}_{\mathsf{ABE}}$ returns message $(C, C_1, \{C_i\}_{i \in S})$. $\mathcal{B}$ then generates the missing component for the ciphertext as

$$C_t = C_1^{r_{u'}} \prod_{i \in \mathcal{V}} C_1^{r_{u_i}} = \left(U' \prod_{i \in \mathcal{V}} U \right)^r.$$

$\mathcal{B}$ returns the ciphertext $(C, C_1, \{C_i\}_{i \in S^*}, C_t)$ to $\mathcal{A}$.

Case 2: $rev = 1$ and for all $t \le t^*$, it has $(id, t) \notin rl$ such that the access structure of id is $\mathbb{A}(S^*) = 1$, then it aborts since all users with $\mathbb{A}(S^*) = 1$ must be revoked before or at the time t^* when $rev = 1$.

Case 3: If $rev = 1$ and for all $t \le t^*$, it has $(id, t) \in rl$ such that the access structure of id is $\mathbb{A}(S^*) = 1$, then $\mathcal{B}$ forwards (m_0, m_1) and t^* to $\mathcal{C}_{\mathsf{IBE}}$. $\mathcal{C}_{\mathsf{IBE}}$ returns the message (C, C_1, C_t).

Recall that $T(i) = g_2^{i^n+u(i)} g^{f(i)} = g^{f(i)}$ for all $\rho(i) \in S^*$. For each $\rho(i) \in S^*$, $\mathcal{B}$ computes the missing component for the ciphertext as

$$C_i = C_1^{f(i)} = \left(g^{f(i)} \right)^r = T(i)^r.$$

$\mathcal{B}$ returns the ciphertext $(C, C_1, \{C_i\}_{\rho(i) \in S^*}, C_t)$ to $\mathcal{A}$.

Phase 2: Same as **Phase 1**.

Guess: $\mathcal{A}$ submits a bit b' to $\mathcal{B}$, then $\mathcal{B}$ forwards b' to $\mathcal{C}_{\mathsf{ABE}}$ if $rev = 0$ and $\mathcal{C}_{\mathsf{IBE}}$ if $rev = 1$.

If $rev = 0$, $\mathcal{B}$ simulates the update key by embedding the key α_x. $\mathcal{C}_{\mathsf{ABE}}$ generates the secret key for the attribute set hiding the key α. $\mathcal{B}$ then transfers this secret key to $\alpha - \alpha_x$. Thus, $\mathcal{A}$ extracts the information $e(g, g)^{\alpha - \alpha_x}$ in the attribute-related component and $e(g, g)^{\alpha_x}$ in the time-related component. This follows our proposal scheme. Additionally, the challenge ciphertext comes from the $\mathcal{C}_{\mathsf{ABE}}$ and $\mathcal{B}$ simulates

the missing time-related component to construct the challenge ciphertext of our proposed scheme. Finally, $\mathcal{B}$ returns a bit b' to $\mathcal{C}_{\mathsf{ABE}}$. $\mathcal{C}_{\mathsf{ABE}}$ then use b' to break decisional BDH problem.

If $rev = 1$, $\mathcal{B}$ simulates differently depending the nodes in the tree BT since $\mathcal{B}$ will answer the key update oracle $\mathcal{O}_{\mathcal{KU}}(\cdot)$ when $\mathcal{A}$ queries the challenge time t^*. If the node is irrelevant to the challenge attribute S^*, $\mathcal{B}$ simulates $\alpha - \alpha_x$ in attribute-related component and α in time-related component. If the node relates to the challenge attribute S^*, $\mathcal{B}$ simulates α_x in attribute-related component. $\mathcal{C}_{\mathsf{IBE}}$ returns the key α in time-related component and $\mathcal{B}$ transfers it to $\alpha - \alpha_x$.

Moreover, the challenge ciphertext comes from $\mathcal{C}_{\mathsf{IBE}}$ and $\mathcal{B}$ simulates the missing attribute-related component to construct the challenge ciphertext of our proposed scheme. Finally, $\mathcal{B}$ returns a bit b' to $\mathcal{C}_{\mathsf{IBE}}$. $\mathcal{C}_{\mathsf{IBE}}$ then utilizes b' to break decisional BDH problem.

There are two aborts during our simulation since $\mathcal{B}$ needs to guess $\mathcal{A}$ is either a non-revoked user or a revoked user. It aborts when $\mathcal{B}$ guessing incorrectly. Thus, the advantage of $\mathcal{A}$ breaks either Goyal *et al.'s* ABE [2] or Waters' IBE [19] is $\epsilon/2$. □

D. Our RABE Scheme in the Random Oracle

Motived by Pirretti *et al.* [13] who proposed an efficient ABE scheme in the random oracle model, we also present our second RABE scheme in the random oracle by utilizing two cryptographic hash functions. More precisely, we replace the function $T(x)$ in the attribute-related component and the public parameters $(U', \{U_i\}_{i\in[\log_2 \mathcal{T}]})$ in the time-related component by two hash functions $H_1 : \{0,1\}^* \to \mathbb{G}$ and $H_2 : \{0,1\}^* \to \mathbb{G}$, respectively.

In the attribute-related component, the function $T(x) = g_2^{x^n} \prod_{i=1}^{n+1} t_i^{\Delta_{i,N}(x)}$ is to generate the message to form the attribute hiding component. In our construction II, the function $T(x)$ is replaced by the hash function H_1 as $T(x) = H_1(x)$.

In the time-related component, the public parameters $(U', \{U_i\}_{i\in[\log_2 \mathcal{T}]})$ have the logarithmic size in the length of the bounded system life time in our construction I. In our construction II, key generation center just publishes a hash function $H_2(x)$ to generate all public parameters for the time-related component by defining $U' = H_2(0)$ and $U_i = H_2(1||0^i)$ for $i \in [\log_2 \mathcal{T}]$, where 0^i denotes a string of i zeros.

Comparing to the our first construction in section IV-B, it significantly reduces the number of elements in the public parameter from logarithmic to constant and also improves computational efficiency.

E. Security Analysis of Our RABE in the Random Oracle Model

In this section, we show that our second RABE construction is also seletively IND-RABE-CPA secure. Due to the similarity between the first construction and the second one, we can prove the security of the latter by following the security proof of the first construction. We omit the detailed description of the simulator and give the sketch of the simulations of the random oracles $H_1(x)$ and $H_2(x)$.

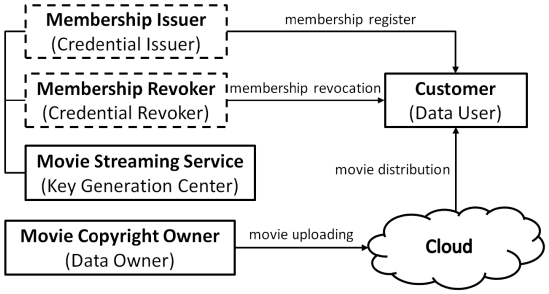

Fig. 3. On-demand Movie Streaming System.

For the private key generation oracle $\mathcal{O}_{\mathsf{SK}}$, the adversary $\mathcal{A}$ sends the message $(id, \mathbb{A})$ to the simulator $\mathcal{B}$. $\mathcal{B}$ keeps a hash list $L = \{(\rho(i), r_i, r_i')\}$. Parse $\mathbb{A} = (M, \rho)$. For each row i in the matrix M: $\rho(i)$ has been queried, $\mathcal{B}$ returns $g_2^{r_i} g^{r_i'}$. Otherwise, if $\rho(i) \in S^*$, $\mathcal{B}$ sets $r_i = 0$ and randomly chooses $r_i' \in \mathbb{Z}_p$; if $\rho(i) \notin S^*$, $\mathcal{B}$ randomly picks $r_i, r_i' \in \mathbb{Z}_p$. Then, $\mathcal{B}$ returns $g_2^{r_i} g^{r_i'}$.

For the key update oracle $\mathcal{O}_{\mathsf{KU}}$, $\mathcal{B}$ randomly chooses $r_i \in \mathbb{Z}_p$ for i from 0 to $\log_2 \mathcal{T}$ and stores them at the beginning of the game. For any queried message t, $\mathcal{B}$ encodes it to $\tilde{t} \leftarrow \mathsf{TEncode}(t, \mathcal{T})$ and defines a set $\mathcal{V}$ for all i of $\tilde{t}[i] = 0$. $\mathcal{B}$ returns $g^{r_0} \prod_{i\in\mathcal{V}} g^{r_i}$.

V. Applications of Our RABE Scheme

Our proposed RABE scheme with ciphertext delegation can enable secure and fine-grained access control in many cloud-based on-demand service applications. The high efficiency of our scheme significantly reduces the workload of the service provider in handling user revocation that occurs frequently in many large-scale applications. In this section, we use the on-demand movie streaming system (depicted in Fig. 3) as an example to demonstrate the usage of our RABE scheme in such applications. Protecting encrypted media (e.g., Videos) in the cloud has been studied in the literature. In [20], a multi-message attribute-based encryption was proposed to enable access control over encrypted media based on the consumers' attributes. In [21], a secure deduplication framework for handling encrypted media in the cloud was introduced to eliminate extra storage and bandwidth cost. In this work, we focus on enabling efficient user revocation for attribute-based cloud media systems.

A cloud-based on-demand movie streaming system consists of the four parties: movie copyright owners (MCOs) such as DreamWorks, a movie streaming service (MSS) such as Amazon Video, a cloud storage provider and the customers. The MSS acts as the KGC and manages (i.e., issues or revokes) the credentials of the customers, while the MCOs act as the data owners and encrypt movies based on their attributes (e.g., movie type, classification, language, duration, region, etc.) and store the encrypted movies on the cloud storage. The customers can subscribe to the KGC by providing some personal information (e.g., age, gender, language, country/region, etc.) and the service period (e.g., free trial for 1 day or a contract for 1 week/month/year). The KGC

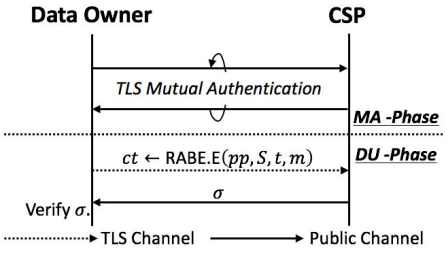

Fig. 4. The Data Owner and the CSP Protocol.

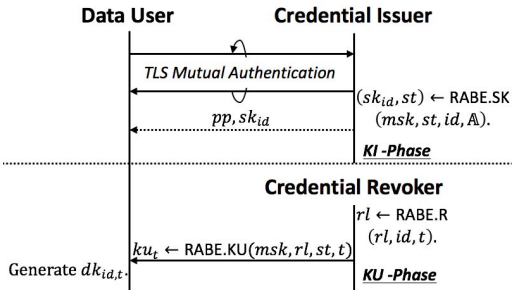

Fig. 5. The Data User and the KGC Protocol.

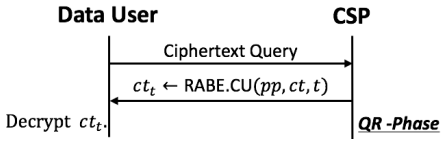

Fig. 6. The Data User and the CSP Protocol.

then generates an access policy and key based on the user information and manages user revocation based on the service period requested by each user.

In our system, we assume the MSS (i.e., KGC) is fully trusted and the cloud storage provider is honest-but-curious (i.e., semi-trusted). That is, we assume the cloud storage provider will generate and send the correct ciphertext corresponding to a time epoch. The customers are untrusted.

Below we describe the protocols implementing our RABE scheme in a cloud-based on-demand movie streaming (or similar) system.

A. Description of the System

The system consists of 3 protocols, namely data uploading, user registration/revocation and data accessing.

Protocol 1: Once a data owner (i.e., MCO) and the service provider (i.e., KGC) have reached an agreement, the data owner will upload the data into the cloud storage. The data uploading protocol consists of two phases, mutual authentication phase (MP-Phase) and data uploading phase (DU-Phase) as shown in Fig. 4.

The MA-Phase can be implemented via a standard TLS handshake initiated by the data owner. It is worth noting that in order to protect the copyright, we should ensure that the data is originated from the legitimate data owner and is uploaded to the correct cloud server. Therefore, mutual authentication is required between the two parties.

In the DU-Phase, the data owner encrypts a movie m using our RABE scheme $ct \leftarrow \mathsf{RABE.E}(pp, S, t, m)$ based on the movie attributes S, such as movie type, classification, region, etc., and the current time t and uploads the ciphertext ct to the cloud storage via the established TLS channel. The cloud storage then signs the received ciphertext and returns the receipt σ to the data owner as a proof that the movie has been uploaded.

Protocol 2. The user registration and revocation (Fig. 5) are performed between the KGC (i.e., MSS) and the data users (i.e., customers). Note that the KGC represents both credential issuer and credential revoker in this application.

The Key Issuing (KI) procedure is performed when a user subscribes to the service. The user first establishes a secure channel with the KGC via TLS and performs registration by providing a unique identifier id, his/her personal information and the service period. The KGC then generates an access policy $\mathbb{A}$ based on the information provided by the data user and produces the corresponding secret key $sk_{id} \leftarrow \mathsf{RABE.SK}(msk, st, id, \mathbb{A})$ that will be sent to the user via the established TLS channel.

The Key Update (KU) procedure is performed at the beginning of a time epoch t. The KGC updates the revocation list by checking the service period of each user and then publishes the update key $ku_t \leftarrow \mathsf{RABE.KU}(msk, rl, st, t)$ to all the users. A non-revoked data user id obtains the updated decryption key $dk_{id,t}$ in revocation epoch t as $dk_{id,t} \leftarrow \mathsf{DK}(sk_{id}, ku_t)$.

Protocol 3. The data accessing (i.e., movie watching) protocol is performed between a data user and the cloud storage as shown in Fig. 6. The protocol starts when the data user sends a data access request to the cloud storage. The cloud storage generates and sends $ct_t \leftarrow \mathsf{RABE.CU}(pp, ct, t)$ to the data user based on the original ciphertext ct uploaded by the MCO and the current time t. The data user decrypts ct_t as $m \leftarrow \mathsf{RABE.D}(pp, dk_{id,t}, ct_t)$.

Remark: In the above protocol, we considered our proposed RABE scheme as a building block. Nevertheless, an issue will arise when the RABE scheme is implemented as a hybrid encryption scheme. That is, we use ABE to encrypt a symmetric-key K, and then encrypt the real data by a symmetric-key encryption scheme (e.g., AES) with K. Under the hybrid encryption setting, if a decryptor has previously decrypted and saved the symmetric-key K, then the decryptor can still use K to access the data even when the ABE ciphertext containing K is updated. We should note that this issue also exists in other approaches (e.g., proxy re-encryption) that allow update of a ciphertext without changing the underlying message. On the other hand, if we purely use ABE to encrypt the real data, then the issue of reusing a symmetric-key won't occur. However, the cost for encrypting and decrypting a large document will significantly increase and updating all the ciphertexts corresponding to the document will also incur a large computation overhead to the CSP under this approach, which may affect the quality of service (e.g., a document is inaccessible during update). Finding an efficient and scalable solution to address this issue is an interesting yet very challenging research problem and we leave it as our future work.

TABLE I

SPACE & COMPUTATIONAL COMPLEXITY OF RABE WITH CIPHERTEXT DELEGATION

Scheme	Space Complexity			Computation Complexity		
	Public Parameter	*Update Key*	*Ciphertext*	*Key Update*	*Encryption*	*Decryption*
SSW [8]	$O(S+\log \mathcal{T})$	$O(\log \mathcal{N} \log \mathcal{T})$	$O(\log \mathcal{T}(S+\log \mathcal{T}))$	$O(\log \mathcal{N} \log \mathcal{T})$	$O(S+log\mathcal{T})$	$O(S+\log \mathcal{T})$
Our Scheme I	$O(S+\log \mathcal{T})$	$O(\log \mathcal{N})$	$O(S+\log \mathcal{T})$	$O(\log \mathcal{N})$	$O(S)$	$O(S)$
Our Scheme II	$O(1)$	$O(\log \mathcal{N})$	$O(S+\log \mathcal{T})$	$O(\log \mathcal{N})$	$O(S)$	$O(S)$

S denotes the maximum size of the attribute set in encryption, $\mathcal{T}$ denotes the bounded system life time, and $\mathcal{N}$ denotes the number of user in the system.

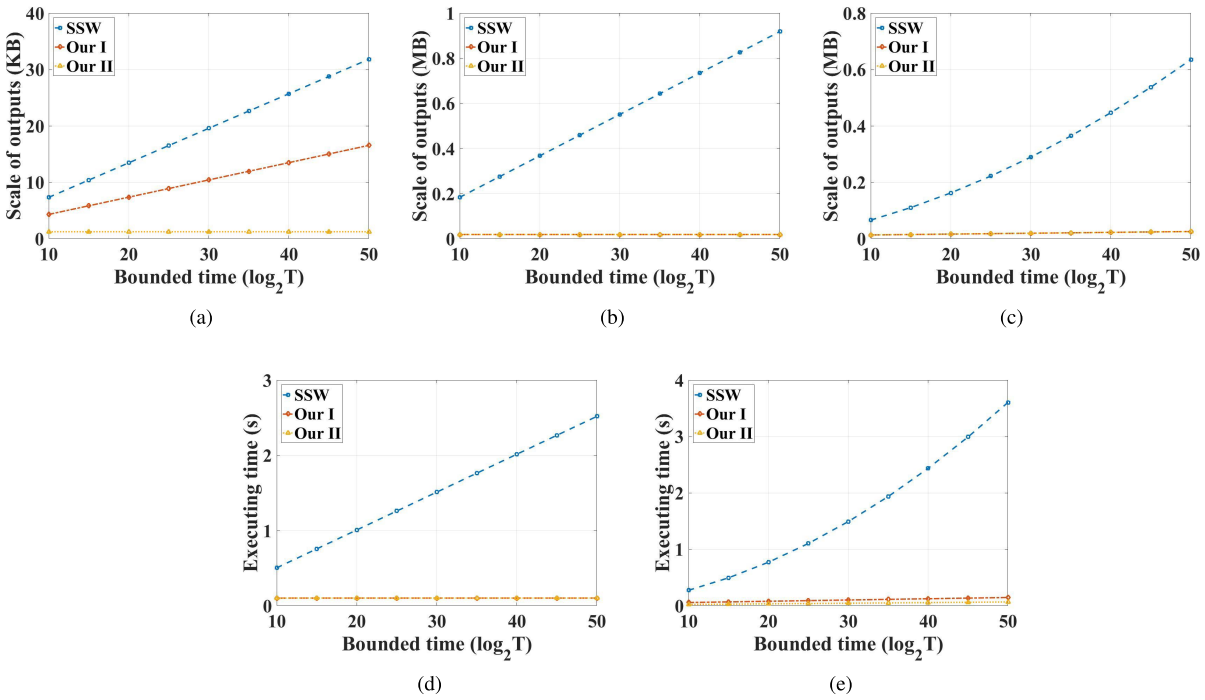

Fig. 7. Experimental results. (a) Size of the public parameter. (b) Size of the update key. (c) Size of the ciphertext. (d) Time of the key update. (e) Time of the encryption.

VI. EXPERIMENTAL RESULTS

In this section, we provide experimental results comparing the RABE scheme in [8] and our newly proposed RABE schemes under different settings. We have implemented the above schemes in C with the PBC library (pbc-0.5.14). In particularly, we have used a Type A elliptic curve $y^2 = x^3 + x$, and the symmetric pairing setting. Additionally, we used the parameters given by "a.param" in PBC library, which produces an elliptic curve bilinear group with 160-bit group order, 512-bit base field and embedded degree 2. Hence, in our scheme, p is a 160-bit prime number, and elements in $\mathbb{G}$ and $\mathbb{G}_T$ have 512 bits and 1024 bits, respectively.

In comparing the three RABE schemes, we set the maximum number of attributes in encryption to 30 and the bounded system life time from 2^{10} to 2^{50}. In each experiment, we set the number of users $\mathcal{N}$ to be 2^{30}, and randomly pick a value $\mathcal{R} < \mathcal{N}$ as the number of revoked users. The software implementation was performed on a desktop running 64-bit Ubuntu 16.04 with 3.40GHz Intel(R) Core(TM) i5-3570 CPU and 8 GB memory. The asymptotic space and computation complexity is given in table I. The experimental results are presented in Fig. 7a–7e. Since the RABE scheme in [8] is based on the composite order groups, we implemented a modified version of the scheme based on Goyal *et al.'s* ABE [2] under prime order groups. To obtain accurate experimental results, for each bounded system life time $\mathcal{T}$, we tested each case for 100 times and calculated the average value.

From Fig. 7 we can see that the experimental results are consistent with what we expected from the asymptotic complexities in Table I. Therefore, our constructions have better performance than the state-of-the-art RABE scheme.

VII. CONCLUSION

Revocable attribute-based encryption (RABE) supporting ciphertext delegation is a useful primitive for enabling secure data sharing via a third-party storage service provider such as cloud storage. In this paper, we revisited the state-of-the-art RABE scheme supporting ciphertext delegation and proposed a new construction paradigm that gives more efficient schemes compared with the previous solution. We provided formal security proofs for our proposed schemes and performed experiments to demonstrate that our new schemes are indeed more efficient than the previous solution. We also presented a fine-grained access control and data sharing system for on-demand services based on the proposed RABE scheme.

ACKNOWLEDGMENT

The authors would like to thank Dr. Stefan Katzenbeisser and the anonymous reviewers for their insightful comments on this paper.

REFERENCES

[1] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proc. Annu. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2005, pp. 457–473.

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

[3] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," in *Proc. NDSS Symp.*, 2005, pp. 1–15.

[4] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *Proc. INFOCOM*, Mar. 2010, pp. 1–9.

[5] S. Yu, C. Wang, K. Ren, and W. Lou, "Attribute based data sharing with attribute revocation," in *Proc. 5th ACM Symp. Inf., Comput. Commun. Secur.*, 2010, pp. 261–270.

[6] K. Yang, X. Jia, and K. Ren, "Attribute-based fine-grained access control with efficient revocation in cloud storage systems," in *Proc. 8th ACM Symp. Inf., Comput. Commun. Secur.*, 2013, pp. 523–528.

[7] 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, Jan. 2013.

[8] A. Sahai, H. Seyalioglu, and B. Waters, "Dynamic credentials and ciphertext delegation for attribute-based encryption," in *Proc. Adv. Cryptol. (CRYPTO)*, 2012, pp. 199–217.

[9] A. B. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters, "Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption," in *Proc. Adv. Cryptol. (EUROCRYPT)*, 2010, pp. 62–91.

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

[11] N. Attrapadung, J. Herranz, F. Laguillaumie, B. Libert, E. de Panafieu, and C. Ràfols, "Attribute-based encryption schemes with constant-size ciphertexts," *Theor. Comput. Sci.*, vol. 422, pp. 15–38, Mar. 2012.

[12] B. Waters, "Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions," in *Proc. Adv. Cryptol. (CRYPTO)*, 2009, pp. 619–636.

[13] M. Pirretti, P. Traynor, P. D. McDaniel, and B. Waters, "Secure attribute-based systems," in *Proc. 13th ACM Conf. Comput. Commun. Secur.*, 2006, pp. 99–112.

[14] D. Boneh and M. K. Franklin, "Identity-based encryption from the Weil pairing," in *Proc. Annu. Int. Cryptol. Conf.*, 2001, pp. 213–229.

[15] A. Boldyreva, V. Goyal, and V. Kumar, "Identity-based encryption with efficient revocation," in *Proc. 15th ACM Conf. Comput. Commun. Secur.*, 2008, pp. 417–426.

[16] N. Attrapadung and H. Imai, "Attribute-based encryption supporting direct/indirect revocation modes," in *Proc. IMA Int. Conf. Cryptogr. Coding*, 2009, pp. 278–300.

[17] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Proc. Int. Workshop Public Key Cryptogr.*, 2011, pp. 53–70.

[18] D. Naor, M. Naor, and J. Lotspiech, "Revocation and tracing schemes for stateless receivers," in *Proc. Annu. Int. Cryptol. Conf.*, 2001, pp. 41–62.

[19] B. Waters, "Efficient identity-based encryption without random oracles," in *Proc. Adv. Cryptol. (EUROCRYPT)*, 2005, pp. 114–127.

[20] Y. Wu, Z. Wei, and R. H. Deng, "Attribute-based access to scalable media in cloud-assisted content sharing networks," *IEEE Trans. Multimedia*, vol. 15, no. 4, pp. 778–788, Jun. 2013.

[21] Y. Zheng, X. Yuan, X. Wang, J. Jiang, C. Wang, and X. Gui, "Toward encrypted cloud media center with secure deduplication," *IEEE Trans. Multimedia*, vol. 19, no. 2, pp. 251–265, Feb. 2017.

Shengmin Xu received the bachelor's and Honor's degrees from the School of Computing and Information Technology, University of Wollongong, Australia, in 2013 and 2014, respectively, where he is currently pursuing the Ph.D. degree, under the supervision of Dr. G. Yang and Prof. Y. Mu. His research interests include cryptography and information security.

Guomin Yang (SM'17) received the Ph.D. degree in computer science from the City University of Hong Kong in 2009. He was a Research Scientist with the Temasek Laboratories, National University of Singapore, from 2009 to 2012. He is currently a Senior Lecturer with the School of Computing and Information Technology, University of Wollongong, Australia. His research mainly focuses on applied cryptography and network security. He received the Australian Research Council Discovery Early Career Researcher Award in 2015.

Yi Mu (SM'03) received the Ph.D. degree from the Australian National University in 1994. He currently is a Professor with the School of Computing and Information Technology, University of Wollongong, Australia. His current research interests include information security and cryptography. He is the Editor-in-Chief of the *International Journal of Applied Cryptography* and serves as associate editor for many other international journals.

Robert H. Deng (F'16) is currently the AXA Chair Professor of Cybersecurity and the Director of the Secure Mobile Centre, School of Information Systems, Singapore Management University (SMU). His research interests are in the areas of data security and privacy, cloud security, and Internet of Things security. He received the Outstanding University Researcher Award from the National University of Singapore, the Lee Kuan Yew Fellowship for Research Excellence from SMU, and the Asia-Pacific Information Security Leadership Achievements Community Service Star from the International Information Systems Security Certification Consortium. His professional contributions include an extensive list of positions in several industry and public services advisory boards, editorial boards, and conference committees. These include the Editorial Boards of *IEEE Security & Privacy Magazine*, the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, the IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, the *Journal of Computer Science and Technology*, and the Steering Committee Chair of the ACM Asia Conference on Computer and Communications Security.