Attribute-Based Encryption with Granular Revocation

Hui Cui, Robert H. Deng(✉), Xuhua Ding, and Yingjiu Li

Secure Mobile Centre, School of Information Systems,
Singapore Management University, Singapore, Singapore
{hcui, robertdeng, xhding, yjli}@smu.edu.sg

Abstract. Attribute-based encryption (ABE) enables an access control mechanism over encrypted data by specifying access policies over attributes associated with private keys or ciphertexts, which is a promising solution to protect data privacy in cloud storage services. As an encryption system that involves many data users whose attributes might change over time, it is essential to provide a mechanism to selectively revoke data users' attributes in an ABE system. However, most of the previous revokable ABE schemes consider how to disable revoked data users to access (newly) encrypted data in the system, and there are few of them that can be used to revoke one or more attributes of a data user while keeping this user active in the system. Due to this observation, in this paper, we focus on designing ABE schemes supporting selective revocation, i.e., a data user's attributes can be selectively revoked, which we call ABE with granular revocation (ABE-GR). Our idea is to utilize the key separation technique, such that for any data user, key elements corresponding to his/her attributes are generated separately but are linkable to each other. To begin with, we give a basic ABE-GR scheme to accomplish selective revocation using the binary tree data structure. Then, to further improve the efficiency, we present a server-aided ABE-GR scheme, where an untrusted server is introduced to the system to mitigate data users' workloads during the key update phase. Both of the ABE-GR constructions are formally proved to be secure under our defined security model.

Keywords: Granular revocation · ABE · Efficiency · Cloud storage

1 Introduction

Attribute-based encryption (ABE) [21] provides a promising solution to preserve data privacy in a scenario (e.g., cloud storage services [23]) where data users are identified by their attributes (or credentials), and data owners want to share their data according to some policy based on the attributes of data users. In a ciphertext-policy ABE (CP-ABE) system, each data user is given a private attribute-key reflecting his/her attributes generated by the attribute authority (AA), and each data owner specifies an access policy to the message over a set of attributes[1]. A data user will be able to decrypt

[1] There are two complimentary forms of ABE: CP-ABE and key-policy ABE (KP-ABE). In a KP-ABE system, the situation is reversed that the keys are associated with the access policies and the ciphertexts are associated with the attributes. In the rest of this paper, unless otherwise specified, what we talk about is CP-ABE.

R. Deng et al. (Eds.): SecureComm 2016, LNICST 198, pp. 165–181, 2017.
DOI: 10.1007/978-3-319-59608-2_9

a ciphertext if and only if the attributes ascribed to his/her private attribute-key satisfy the access policy (or access structure) associated with the ciphertext. Though ABE favorably solves the problem arising in the situations where different users with different attributes are given access to different levels of the encrypted data, it fails to address the issue of dynamic credentials where the attributes of every data user change with time. This challenge motivates the study of revocation mechanisms [1], where a periodical key update process disables revoked data users to update their decryption keys to decrypt newly encrypted data.

In terms of attribute-based setting, user revocation is divided into indirect revocation and direct revocation [1]. Regarding indirect revocation, one solution is to ask data users to periodically renew their private attribute-keys [7], but this requires the update key size to be $O(N - R)$ group elements where N is the number of all users and R is the number of revoked users. To reduce the cost of key update from linear to logarithmic $\left(\text{i.e., } O\left(R \log\left(\frac{N}{R}\right)\right)\right)$, Boldyreva, Goyal and Kumar [5] put forth a revocation methodology by combining the fuzzy IBE scheme [21] with the binary tree data structure [18] where the AA publicly broadcasts the key update information for each time period, but only non-revoked data users can update their decryption keys to decrypt a newly generated ciphertext. In direct revocation [1, 2], data owners possess a current revocation list, and specify the revocation list directly when running the encrypting algorithm so that user revocation can be done instantly without requiring the key update phase as in the indirect method[2]. There are also constructions (e.g., [25]) that delegate the direct revocation ability to a semi-trusted server who cannot collude with the data users, where the server helps data users with decryption but terminates the decryption operation for any revoked data user.

Since an attribute-based encryption system might involve a large number of data users whose attributes change over time, it is desirable to build an attribute-based encryption scheme that the credentials possessed by data users can be selectively revoked. However, most of the previous revocable ABE systems [1, 2, 5, 20, 25] only consider efficient user revocation to prevent revoked data users from accessing the encrypted data, and there is little attention on how to independently revoke one or more attributes from a data user, i.e., selective revocation on attributes. Due to this observation, in this paper, we focus on the design of efficient and revocable attribute-based encryption schemes where the attributes possessed by each data user can be selectively revoked via a periodical key update phase, which we call attribute-based encryption with granular revocation (ABE-GR). Notice that ABE-GR can achieve user revocation by revoking all credentials possessed by a data user.

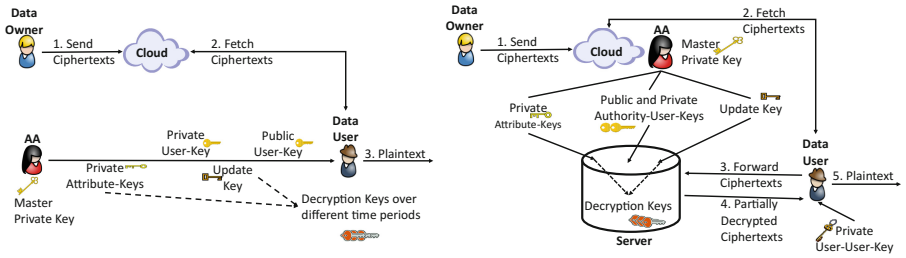
[2] Note that direct revocation can be done immediately without the key update process which asks for the communication from the AA to all the non-revoked users over all the time periods, but it requires all the data owners to keep the current revocation list. This makes the system impurely attribute-based, since data owners in the attribute-based setting create ciphertext based only on attributes without caring revocation. In this paper, unless otherwise specified, the revocation mechanism we talk about is indirect revocation.

1.1 Our Contributions

We describe the system architecture of an ABE-GR scheme in Fig. 1-(a). In an ABE-GR system, each data user's key is divided into three parts: private user-key (with a corresponding public user-key), private attribute-keys (associated with different attributes) and public key update information, from the latter two of which a data user can extract a decryption key. The AA, who keeps the master private key and publishes the public parameter, is responsible for the distribution of personalized pairs of public and private user-keys and private attribute-keys. In addition, the AA regularly posts the public key update information. Each data owner encrypts a message under an access structure and a time period using the public parameter. To decrypt a newly generated ciphertext, a data user needs to possess a pair of public and private user-keys as well as a decryption key on the current time period satisfying the access policy of this ciphertext. The key challenge in building an ABE-GR scheme is to prevent a data user from using his/her revoked attributes to decrypt any newly generated ciphertext. Traditionally, in an ABE scheme, each attribute possessed by a data user corresponds to one element in his/her private attribute-key, and these key elements are tied together through a random value. In order to support granular revocation in ABE, we need a technique to enable different key components on different attributes to be created separately but linkable to each other. Thanks to the key separation technique in distributed ABE [17] where the task of the single AA is split across multiple AAs and each attribute is controlled by one specific AA, we can equip an ABE scheme with a similar technique but under a single AA. Thus, each key component associated with the corresponding attribute will be created separately, but they still bind together due to the sharing of the same identification information (i.e., the public user-key) which is unique to each data user. As a result, we build an ABE-GR scheme by combining an ABE scheme with the key separation technique in distributed ABE [17]. To reduce the size of key update for the AA from linear to logarithmic in the number of users, we apply the binary tree data structure [18] in the algorithms of our ABE-GR scheme. Details about this ABE-GR scheme, which we will refer to as a basic ABE-GR scheme, is given in Sect. 4.

As alluded in [19], binary tree data structure [18] is useful in alleviating the workload of the AA, but it could not mitigate the workload of each data user who needs to periodically update the decryption key. Is it possible to fix the keys stored by data users such that they are not required to frequently update their decryption keys while without affecting the revocation? To give an affirmative answer to this question, we bring in an untrusted server[3] to the basic ABE-GR system to mitigate the workloads of data users. We depict the system architecture in Fig. 1-(b), which involves four entities: an AA, data owners, data users, and a server. Different from that in the basic ABE-GR construction, the public and private user-key pair is divided into a pair of public and private user-user-keys and a pair of public and private authority-user-keys, of which the

[3] The server is untrusted in the sense that it honestly follows the protocol but without holding any secret information (i.e., it may collude with data users). Besides, all operations done by the server can be performed by anyone, including data users (i.e., any dishonest behaviour from the server can be easily detected).



(a) ABE-GR (b) Server-Aided ABE-GR

Fig. 1. System architectures of ABE-GR (left) and server-aided ABE-GR (right).

former is generated by each data user himself/herself[4] and the latter is extracted by the AA based on the public user-user-key. The server is given the public and private authority-user-keys and private attribute-keys of data users as well as the key update information. A data user fetches a ciphertext from the cloud, and sends it to the server for partial decryption. For any non-revoked user, from the private attribute-keys and key update information, the server can generate a collection of decryption keys (associated with a set of attributes), which, combining with the public and private authority-user-keys, can partially decrypt a ciphertext forwarded by this user if his/her non-revoked attributes satisfy the access structure ascribed to the ciphertext. A data user can obtain the plaintext by decrypting the partially decrypted ciphertext using his/her self-generated private user-user-key. This does not compromise the security, because the public user-user-key is embedded in the private authority-user-key, the server cannot fully unwrap the ciphertext without the private user-user-key. A detailed description of the construction is presented in Sect. 4.

Since both our constructions are built on an ABE scheme that is selectively secure [17, 24], where the adversary has to commit the challenge access structure in advance, we can only achieve selective security in our ABE-GR schemes. Note that the techniques can be applied to fully secure ABE schemes (e.g., [20]) to obtain fully secure (server-aided) ABE-GR schemes.

1.2 Related Work

Revocable IBE. With regard to revocable IBE, Boneh and Franklin [7] suggested that users periodically renew their private keys, but this method has a disadvantage in that it requires all users to regularly contact the key generation centre (KGC) to obtain new private keys, and thus a secure channel must be established between the KGC and each user for such transactions. Hanaoka et al. [10] presented a convenient methodology for users to periodically renew their private keys without communicating with the KGC by making the KGC publicly post the key update information. However, as each user in

[4] This pair of user-user-keys can also be generated by the AA, but this requires a secure channel between each data user and the AA for private key distribution.

this case needs to posses a tamper-resistant hardware device, the solution is very cumbersome. Boldyreva, Goyal and Kumar [5] put forth an efficient revocable IBE scheme to reduce the size of key update from linear to logarithmic, but it asks all non-revoked users to regularly update their decryption keys. To address the revocation issue in a better way, revocation with a third party [3, 6, 9, 13, 15, 16, 19] has been introduced, in which a semi-trusted[5] (or untrusted) third party is assigned to share the decryption capability with all users and help them the ciphertext decryption. Once an identity is revoked, the mediator immediately terminates decrypting the ciphertext for this user.

Revocable ABE. Regarding user revocation in ABE, there are two revocation mechanisms [1, 8]: direct revocation and indirect revocation. Considering the former, in which each data owner keeps a current revocation list, and directly specify the revocation list when encrypting, there are schemes in [2, 11, 14]. In addition, Yang et al. [25] put forward an approach by assigning a semi-trusted server to share the decryption capability with data users such that when a data user is revoked, the server stops the decryption for the user. Regarding the latter, which we intend to achieve in this paper, Boldyreva, Goyal and Kumar [5] proposed a revocable KP-ABE scheme where the AA indirectly achieves the revocation by disabling revoked users to update their keys. Later, based on the same technique adopted in [5], Attrapadung and Imai [1] gave a hybrid revocable KP-ABE scheme under selective security model which allows a data owner to select the revocation mode (direct or indirect) when performing encryption. Sahai, Seyalioglu and Waters [20] showed a generic way to build ABE schemes that support dynamic credentials, where the AA indirectly accomplishes revocation by stopping updating the keys for revoked users. Cui and Deng [8] proposed two indirectly revocable and decentralized ABE schemes in the composite-order groups where the AA'role is split over multiple AAs.

1.3 Organization

The remainder of this paper is organized as follows. In Sect. 2, we briefly review the relevant notions and definitions to be used in this paper. In Sect. 3, after describing the framework for ABE-GR, we present its security model. In Sect. 4, we present two concrete constructions of ABE-GR, and provably reduce their security. In addition, we compare our ABE-GR schemes with previous revocable ABE schemes in Sect. 4. We conclude the paper in Sect. 5.

2 Preliminaries

In this section, we review some basic cryptographic notions and definitions that are to be used in this paper.

[5] In this paper, unless otherwise specified, "semi-trusted" means that the corresponding entity is disallowed to collude with the malicious data users.

2.1 Bilinear Pairings and Complexity Assumptions

Let p be a prime number, and G be a group of order p that is generated from g. We define $\hat{e} : G \times G \rightarrow G_1$ to be a bilinear map if it has two properties [7].

- Bilinear: for all $g \in G$, and $a, b \in Z_p$, we have $\hat{e}(g^a, g^b) = \hat{e}(g, g)^{ab}$.
- Non-degenerate: $\hat{e}(g, g) \neq 1$.

We say that G is a bilinear group if the group operation in G is efficiently computable and there exists a group G_1 and an efficiently computable bilinear map $\hat{e} : G \times G \rightarrow G_1$ as above.

Decisional Parallel Bilinear Diffie-Hellman Exponent Assumption [24]. The decisional q-parallel bilinear Diffie-Hellman exponent (BDHE) problem is that for any probabilistic polynomial-time (PPT) algorithm, given

$$\overrightarrow{y} = \begin{array}{ll} \forall j \in [1, q] & g, g^{\mu}, g^{a}, \ldots, g^{a^q}, g^{a^{q+2}}, \ldots, g^{a^{2q}}, \\ & g^{\mu \cdot b_j}, g^{a/b_j}, \ldots, g^{a^{q}/b_j}, g^{a^{q+2}/b_j}, \ldots, g^{a^{2q}/b_j}, \\ \forall 1 \leq j, k \leq q, k \neq j & g^{a \cdot \mu \cdot b_k / b_j}, \ldots, g^{a^q \cdot \mu \cdot b_k / b_j}, \end{array}$$

it is difficult to distinguish $\left(\overrightarrow{y}, \hat{e}(g, g)^{a^{q+1}\mu}\right)$ from $\left(\overrightarrow{y}, Z\right)$, where $g \in G, Z \in G_1, a$, $\mu, b_1, \ldots, b_q \in Z_p$ are chosen independently and uniformly at random.

2.2 Access Structures and Linear Secret Sharing

Definition 1 ***(Access Structure** [12, 24]). Let $\{P_1, \ldots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1, \ldots, P_n\}}$ 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 $\{P_1, \ldots, P_n\}$, i.e., $\mathbb{A} \subseteq 2^{\{P_1, \ldots, P_n\}} \backslash \{\emptyset\}$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called the unauthorized sets.*

Definition 2 ***(Linear Secret Sharing Schemes (LSSS)** [12, 24]). Let P be a set of parties. Let $\mathbb{M}$ be a matrix of size $l \times n$ (l rows and n columns). Let $\rho : \{1, \ldots, l\} \rightarrow P$ be a function mapping a row to a party for labeling. A secret sharing scheme Π over a set of parties P is a linear secret-sharing scheme over Z_p if*

1. *The shares for each party form a vector over Z_p.*
2. *There exists an $l \times n$ matrix $\mathbb{M}$ named the share-generating matrix for Π. For $x = 1, \ldots, l$, the x-th row of matrix $\mathbb{M}$ is labeled by a party $\rho(i)$ for $\rho : \{1, \ldots, l\} \rightarrow P$ being a function mapping a row to a party for labeling. Considering that the column vector $\overrightarrow{v} = (\mu, r_2, \ldots, r_n)$ with $\mu \in Z_p$ being the secret to be shared and $r_2, \ldots, r_n \in Z_p$, then $\mathbb{M}\overrightarrow{v}$ is the vector of l shares of the secret μ according to Π. The share $(\mathbb{M}\overrightarrow{v})_i$ belongs to party $\rho(i)$.*

It has been concluded in [12] that every LSSS enjoys a property called linear reconstruction. Assume that Π is an LSSS for an access structure A. Denote $\mathbf{A}$ as an authorized set, and $I \subseteq \{1, \ldots, l\}$ as $I = \{i | \rho(i) \in \mathbf{A}\}$. Then the vector $(1, 0, \ldots, 0)$ is in the span of rows of matrix $\mathbb{M}$ indexed by I, and there exist constants $\{w_i \in Z_p\}_{i \in I}$ such that, for any valid shares $\{v_i\}$ of a secret μ according to Π, $\sum_{i \in I} w_i v_i = \mu$ holds. These constants $\{w_i\}$ can be found in polynomial time depending on the size of the share-generating matrix $\mathbb{M}$ [4].

Boolean Formulas [12]. An access policies can be described in a monotonic boolean formula as well. An LSSS access structure is more general, and can be derived from a representation as a boolean formula. There are generic techniques to transfer any monotonic boolean formula into an LSSS matrix. An boolean formula can be represented as an access tree with the interior nodes being AND and OR gates and the leaf nodes corresponding to attributes. Note that the number of rows in the corresponding LSSS matrix is the same as the number of leaf nodes in the access tree.

2.3 Terminologies on Binary Tree

We follow the definitions about the binary tree in [5, 19]. Denote BT as a binary tree with N leaves representing N users, and **root** as the root node of the tree BT. Let Path (θ) be the set of nodes on the path from θ to **root** (which includes both θ and **root**) if θ is a leaf node. Let θ_l and θ_r be left and right child of θ if θ is a non-leaf node. Assume that nodes in the tree are uniquely encoded as strings, and the tree is defined by all descriptions of its nodes. There is an algorithm KUNodes defined to calculate the minimal set of nodes for which the key update needs to be published so that only non-revoked users at a time period t can decrypt ciphertexts, which works by firstly marking all the ancestors of the revoked nodes as revoked, and then outputting all the non-revoked children of revoked nodes. Formally, the KUNodes algorithm takes a binary tree BT, a revocation list rl and a time period t as the input, and outputs a set of nodes, the minimal set of nodes in BT, such that none of the nodes in rl with the corresponding time period at or before t (users revoked at or before t) have any ancestor (or, themselves) in the set, and all other leaf nodes (corresponding to non-revoked users) have exactly one ancestor (or, themselves) in the set.

KUNodes (BT, rl, t)
$X, Y \leftarrow \varnothing$.
$\forall\ (\theta_i, t_i) \in rl$, if $t_i \leq t$, then add Path(θ_i) to X.
$\forall\ x \in X$, if $x_l \notin X$, then add x_l to Y; if $x_r \notin X$, then add x_r to Y.
If $Y = \varnothing$, then add **root** to Y.
Return Y.

3 System Architecture and Security Definition

We describe the system architecture and formal security definition of attribute-based encryption with granular revocation in this section.

3.1 Framework

An ABE-GR scheme involves three entities: attribute authority (AA), data owners and data users, where the algorithms run by these parties are described as follows.

- GSetup(1^λ) $\rightarrow$ (par, msk). Taking a security parameter λ as the input, this algorithm, run by the AA, outputs the public parameter par and the master private key msk.
- ASetup(par, A_i) $\rightarrow$ ($\mathrm{PK}_{A_i}, \mathrm{SK}_{A_i}$, rl_i, st_i). Taking the public parameter par and an attribute A_i as the input, this algorithm, run by the AA, outputs a public key PK_{A_i} a private key SK_{A_i} an initially empty revocation list rl_i and a state st_i.
- UserKG(par, msk, id) $\rightarrow$ (sk_{id}, pk_{id}). Taking the public parameter par, the master private key msk and an identity id as the input, this algorithm, run by the AA, outputs a private user-key sk_{id} and a public user-key pk_{id} for user id.
- PrivKG(par, SK_{A_i}, pk_{id}, st_i) $\rightarrow$ (pk_{id}^{Ai}, st_i). Taking the public parameter par, the private key SK_{A_i} a pubic user-key pk_{id} and a state st_i as the input, this algorithm, run by the AA, outputs a private attribute-key pk_{id}^{Ai} and an updated state st_i for user id possessing an attribute A_i.
- TKeyUp(par, SK_{A_i}, t, rl_i, st_i) $\rightarrow$ ($ku_t^{(i)}$, st_i). Taking the public parameter par, the private key SK_{A_i} a time period t, a revocation list rl_i and a state st_i as the input, this algorithm, run by the AA, outputs the key update information $ku_t^{(i)}$ and an updated state st_i.
- DecKG(par, $pk_{id}^{A_i}$, $tku_t^{(i)}$) $\rightarrow$ $dk_{id,t}^{(i)}$. Taking the public parameter par, a private attribute-key $pk_{id}^{A_i}$ and the key update information $tku_t^{(i)}$ as the input, this algorithm, run by each data user, outputs a decryption key $dk_{id,t}^{(i)}$ for used id at time period t.
- Encrypt(par, $(\mathbb{M}, \rho)$, t, $\{\mathrm{PK}_{A_i}\}, M$) $\rightarrow$ CT. Taking the public parameter par, an access structure $(\mathbb{M}, \rho)$, a time period t, a set of public keys $\{\mathrm{PK}_{A_i}\}$ for relevant attributes and a message M as the input, this algorithm, run by each data owner, outputs a ciphertext CT (will be stored to the cloud).
- Decrypt(par, pk_{id}, sk_{id}, $\{dk_{id,t}^{(i)}\}$, CT) $\rightarrow$ $M/\perp$. Taking the public parameter par, a public user-key pk_{id}, a private user-key sk_{id}, a collection of decryption keys $\{dk_{id,t}^{(i)}\}$ of the same id and a ciphertext CT as the input, this algorithm, run by each data user, outputs a message M if the attribute set $\{A_i\}$ satisfies the access matrix associated with the ciphertext or a failure symbol $\perp$.
- Revoke(id, A_i, t, rl_i, st_i) $\rightarrow$ rl_i. Taking an attribute A_i of identity id to be revoked, a time period t, a revocation list rl_i and a state st_i as the input, this algorithm, run by the AA, outputs an updated revocation list rl_i.

Note that in order to create public and private keys, private attribute-keys, key update information and decryption keys corresponding to multiple attributes, the corresponding algorithms ASetup, PrivKG, TKeyUp and DecKG are extended to take in many attributes by running the "single attribute" version once for each attribute.

For the correctness of an ABE-GR scheme, we require that for any security parameter λ and any message M (in the message space), if the data user is not revoked

at time period t (in the space of time periods), and if all entities follow the above algorithms as described, then Decrypt $(par, sk_{id}, \{dk_{id,t}^{(i)}\}, \mathrm{CT}) = M$ if $\{dk_{id,t}^{(i)}\}$ is a set of decryption keys for the same identity id over a set of attributes satisfying the access structure of the ciphertext CT.

3.2 Security Definition

Below we describe the security game between an adversary algorithm $\mathcal{A}$ and a challenger algorithm $\mathcal{B}$ as indistinguishability under chosen plaintext attacks (IND-CPA security) for an ABE-GR scheme.

- Setup. Algorithm $\mathcal{B}$ runs the GSetup algorithm, sends algorithm $\mathcal{A}$ the public parameter par and keeps the master private key msk. Also, it runs the ASetup algorithm, keeps the private keys $\{\mathrm{SK}_{A_i}\}$, initially empty revocation lists $\{rl_i\}$ and states $\{st_i\}$ and sends the public keys $\{\mathrm{PK}_{A_i}\}$ to algorithm $\mathcal{A}$.
- Phase 1. Algorithm $\mathcal{A}$ adaptively issues queries to the following oracles.
  1. Private-User-Key oracle. Algorithm $\mathcal{A}$ issues a private user-key query to algorithm $\mathcal{B}$ on an identity id, and algorithm $\mathcal{B}$ returns sk_{id} by running UserKG(par, msk, id), and adds (id, pk_{id}) to the user list.
  2. Private-Attribute-Key oracle. Algorithm $\mathcal{A}$ issues a public attribute-key query to algorithm $\mathcal{B}$ on an identity id with an attribute set $\{A_i\}$, and algorithm $\mathcal{B}$ returns $\{pk_{id}^{Ai}\}$ by running UserKG(par, msk, id) (if id does not exist in the user list), PrivKG $(par, \mathrm{SK}_{A_i}, pk_{id}, st_i)$ for each A_i of id.
  3. Key-Update oracle. Algorithm $\mathcal{A}$ issues a key update query to algorithm $\mathcal{B}$ on a time period t, and algorithm $\mathcal{B}$ returns $\{tku_t^{(i)}\}$ by running TKeyUp $(par, \mathrm{SK}_{A_i}, t, rl_i, st_i)$ for each A_i.
  4. Decryption-Key oracle. Algorithm $\mathcal{A}$ issues a decryption key query to algorithm $\mathcal{B}$ on a time period t and an identity id with an attribute set $\{A_i\}$, and algorithm $\mathcal{B}$ returns $\{dk_{id,t}^{(i)}\}$ by running UserKG(par, msk, id) (if id does not exist in the user list), PrivKG $(par, \mathrm{SK}_{A_i}, pk_{id}, st_i)$, TKeyUp $(par, \mathrm{SK}_{A_i}, t, rl_i, st_i)$, DecKG $(par, pk_{id}^{A_i}, tku_t^{(i)})$ on each A_i of id. Notice that queries on a time period t that has not been issued to the Key-Update oracle cannot be issued to this oracle.
  5. Revocation oracle. Algorithm $\mathcal{A}$ issues a revocation query to algorithm $\mathcal{B}$ on an attribute A_i of identity id and a time period t, and algorithm $\mathcal{B}$ runs Revoke(id, t, rl_i, st_i) and outputs an updated revocation list rl_i. If a key update query has been issued on a time period t, this oracle cannot be queried on t.
- Challenge. Let $I_{\mathbb{M}^*,\rho^*} = \{\mathcal{I}_1, \ldots \mathcal{I}_\chi$ be a set of minimum subsets of attributes satisfying $(\mathbb{M}^*, \rho^*)$. Algorithm $\mathcal{A}$ outputs two messages M_0^* and M_1^* of the same size, an access structure $(\mathbb{M}^*, \rho^*)$ and a time period t^* following the constraint that for each id, if algorithm $\mathcal{A}$ asks for a collection of private attribute-keys on an attribute set covering an $\mathcal{I}_j \in I_{\mathbb{M}^*,\rho^*}$ $(j \in [1, \chi])$ then (1) the revocation oracle must be queried on some tuple (id, t, A_i) where t happens at or before t^* and $A_i \in \mathcal{I}_j$ $(j \in [1, \chi])$, and (2) the Decryption- Key oracle cannot be queried on (id, t,

$\{A_i\}$) for any $t = t^*$ and $\mathcal{I}_j \subseteq \{A_i\} (j \in [1, \chi])$. Algorithm $\mathcal{B}$ randomly chooses $\gamma \in \{0,1\}^*$, runs Encrypt $(par, (\mathbb{M}^*, \rho^*), \{\mathrm{PK}_{Ai}\}, t^*, M^*_\gamma)$ to obtain the challenge ciphertext CT^*, and sends CT^* to algorithm $\mathcal{A}$.

- Phase 2. Following the restriction defined in the Challenge phase, algorithm $\mathcal{A}$ continues issuing queries to algorithm $\mathcal{B}$ as that in Phase 1.
- Guess. Algorithm $\mathcal{A}$ makes a guess γ' for γ. It wins the game if $\gamma' = \gamma$.

Algorithm $\mathcal{A}$'s advantage in the above game is defined to be $\Pr[\gamma{=}\gamma'] - 1/2$. An ABE-GR scheme is said to be IND-CPA secure under the defined security model if all PPT adversaries have at most a negligible advantage in the security parameter λ λ. In addition, an ABE-GR scheme is said to be selectively IND-CPA secure if an Init stage where algorithm $\mathcal{A}$ commits to the challenge access structure (M^*, ρ^*) which it attempts to attack is added before the Setup phase.

Remarks. Note that the above security definition is different from those in previous revocable ABE schemes. The definitions in [1, 8, 20] did not consider a realistic threat called decryption key exposure attacks [22][6], while the above model allows an additional Decryption-Key oracle to resist such attacks so that no information of the plaintext is revealed from a ciphertext even if all (short-term) decryption keys of different time periods are exposed.

4 Attribute-Based Encryption with Granular Revocation

In this section, we present two ABE-GR constructions and their security analysis. Also, we compare them with several existing revocable ABE schemes.

4.1 Basic Construction

Let the attribute space be Z_p, the time space be Z_p, and the message space be G_1. The basic attribute-based encryption scheme supporting granular revocation is composed of the following algorithms, which is built upon the CP-ABE scheme presented in [24].

- GSetup. On input a security parameter λ, it randomly chooses a group G of a prime order p with $g \in G$ being the generator, and defines a bilinear map $\hat{e} : G \times G \rightarrow G_1$ Additionally, it randomly chooses u, $h \in G$, a, $\alpha \in Z_p$, and defines a function $F(y) = u^y h$ to map an element y in Z_p to an element in G. The public parameter is $par = (g, g^a, u, h, \hat{e}(g,g)^\alpha)$. The master private key is $msk = \alpha$.
- ASetup. On input the public parameter par and an attribute A_i, it randomly chooses $\alpha_i \in Z_p$, and computes $\mathrm{PK}_{A_i} = g^{\alpha_i}$. Let rl_i be an empty list storing revoked users and BT_i be a binary tree with at least N leaf nodes. It outputs the public key PK_{A_i} along with rl_i and st_i where st_i is a state which is set to be BT_i, and keeps α_i as the private key SK_{A_i}.

[6] This does not affect the security of these schemes, because such attacks are not covered by their security models.

- UserKG. On input the public parameter *par*, the master private key *msk* and an identity *id*, it randomly chooses $\beta \in Z_p$, and outputs a private user-key $sk_{id} = g^{\alpha}(g^{a})^{\beta}$ and a public user-key $pk_{id} = g^{\beta}$.
- PrivKG. On input the public parameter *par*, a private key SK_{A_i} a public user-key pk_{id} and a state st_i, it firstly chooses an undefined leaf node $\theta^{(i)}$ from the binary tree BT_i, and stores *id* in this node. Then, for each node $x^{(i)} \in \mathrm{Path}(\theta^{(i)})$, it runs as follows.
  1. It obtains $g_{i,x}$ from the node $x^{(i)}$. If $x^{(i)}$ is undefined, it randomly chooses $g_{i,x} \in G$, and computes $P_x^{(i)} = (g^{\beta}/g_{i,x})^{\alpha_i}$. It stores $g_{i,x}$ in the node $x^{(i)}$.
  2. It outputs the private attribute-key $pk_{id}^{A_i} = \{x^{(i)}, P_x^{(i)}\}_{x^{(i)} \in \mathrm{path}(\theta^{(i)})}$, and an updated state st_i.
- TKeyUp. On input the public parameter *par*, a private key SK_{A_i} a time period *t*, a revocation list rl_i and a state BT_i, for all $x^{(i)} \in \mathrm{KUNodes}(\mathrm{BT}_i, rl_i, t)$, it gets $g_{i,x}$[7] from the node *x*. Then, it randomly chooses $s_{i,x} \in Z_p$, and computes $Q_{x,1}^{(i)} = g_{i,x}^{\alpha_i} \cdot F(t)^{s_{i,x}}$, $Q_{x,2}^{(i)} = g^{s_{i,x}}$. It outputs $ku_t^{(i)} = \{x^{(i)}, Q_{x,1}^{(i)}, Q_{x,2}^{(i)}\}_{x^{(i)} \in \mathrm{KUNodes}(\mathrm{BT}_i, rl_i, t)}$ as the key update information.
- DecKG. On input the public parameter *par,* a private attribute-key pk_{id}^{Ai} and the key update information $tku_t^{(i)}$ as the input, it parses each pk_{id}^{Ai} as $\{x^{(i)}, P_x^{(i)}\}_{x^{(i)} \in I, tku_t^{(i)}}$ as $\{x^{(i)}, Q_{x,1}^{(i)}, Q_{x,2}^{(i)}\}_{x^{(i)} \in J}$ for some set of nodes $I = \mathrm{Path}(\theta^{(i)}), J = \mathrm{KUNodes}(\mathrm{BT}_i,\ rl_i,\ t)$. If $I \cap J = \emptyset$, it returns $\perp$. Otherwise, for any $x^{(i)} \in I \cap J$, it randomly chooses $s'_{i,x} \in Z_p$, and computes

$$dk_1^{(i)} = P_x^{(i)} \cdot Q_{x,1}^{(i)} \cdot F(t)^{s'_{i,x}} = pk_{id}^{\alpha_i} \cdot F(t)^{s_{i,x} + s'_{i,x}},$$
$$dk_2^{(i)} = Q_{x,2}^{(i)} \cdot g^{s'_{i,x}} = g^{s_{i,x} + s'_{i,x}}.$$

  It outputs the decryption key $dk_{id,t}^{(i)} = (dk_1^{(i)}, dk_2^{(i)})$.
- Encrypt. On input the public parameter *par*, an LSSS access structure $(\mathbb{M}, \rho)$ with $\mathbb{M}$ being an $l \times n$ matrix, a set of public keys $\{\mathrm{PK}_{A_i}\}$ for relevant attributes, a time period *t* and a message *M,* it randomly chooses a vector $\overrightarrow{v} = (\mu, y_2, \ldots, y_n)^{\perp} \in Z_p^n$ (these values will be used to share the encryption exponent μ). For $i = 1$ to l, it calculates $v_i = \mathbb{M}_i \cdot \overrightarrow{v}$ where $\mathbb{M}_i$ is the *i*-th row of $\mathbb{M}$. Also, it randomly chooses $\mu, \mu_1, \ldots, \mu_k \in Z_p$, and computes

$$C_0 = \hat{e}(g,g)^{\alpha\mu} \cdot M, \quad C_2^{(i)} = (g^{a})^{v_i} \cdot (\mathrm{PK}_{\rho(i)})^{-\mu_i},$$
$$C_1 = g^{\mu}, \quad C_3^{(i)} = g^{\mu_i}, \quad C_4^{(i)} = F(t)^{\mu_i}.$$

- It outputs the ciphertext $\mathrm{CT} = ((\mathbb{M}, \rho), t, C_0, C_1, \{C_2^{(i)}, C_3^{(i)}, C_4^{(i)}\}_{i \in [i,l]})$.

[7] Here *gi,x* is always predefined in the PrivKG algorithm.

– Decrypt. On input the public parameter *par*, a public user-key pk_{id}, a private user-key sk_{id}, a set of decryption keys $\{dk_{id,t}^{(i)}\}$ and a ciphertext CT, it computes

$$\frac{\hat{e}(C_1,\, sk_{id}) \prod_{i \in I} \hat{e}(C_4^{(i)}, dk_2^{(i)})}{(\prod_{i \in I} \hat{e}(C_2^{(i)}, pk_{id}) \hat{e}(C_3^{(i)}, dk_1^{(i)}))^{w_i}} = \hat{e}(g,g)^{\alpha\mu},$$

and then cancels out this value from C_0 to obtain the plaintext M. Suppose that $\{A_i\}$ associated with $\{pk_{id}^{Ai}\}$ satisfies the access structure $(\mathbb{M}, \rho)$. Let I be defined as $I = \{i : \rho(i) \in \{A_i\}\}$. Denote $\{w_i \in Z_p\}_{i \in I}$ as a set of constants such that if $\{v_i\}$ are valid shares of any secret μ according to $\mathbb{M}$, then $\sum_{i \in I} w_i v_i = \mu$.
– Revoke. On input an attribute A_i of identity *id*, a time period *t*, a revocation list rl_i and a state st_i, for all the nodes $x^{(i)}$ associated with identity *id*, it adds $(x^{(i)}, t)$ to rl_i, and outputs the updated rl_i.

Theorem 1. *Under the decisional q-parallel BDHE assumption, the above basic ABE-GR scheme is selectively IND-CPA secure.*

Proof. In the proof, it is assumed that if an adversary has issued a private user-key query on an identity *id*, and a private attribute-key query on attributes $\{A_i\}$ of this identity *id* satisfying the challenge access structure $(\mathbb{M}^*, \rho^*)$, then at least one attribute in each set of minimum attributes satisfying $(\mathbb{M}^*, \rho^*)$ of this identity *id* is revoked at or before the challenge time period t^*. We detail the proof in the full version of this paper due to the space limit[8].

4.2 Construction with Improved Efficiency

The main drawback in our previous construction lies in that all non-revokes data users need to periodically update their decryption keys. To remove such cumbersome workloads from data users, we give another ABE-GR scheme, which we call a server-aided ABE-GR scheme. Our method is to introduce an untrusted server to the basic ABE-GR scheme such that the server will help data users with the workloads in key update stage. The algorithms of our server-aided ABE-GR scheme mostly follow those in the basic ABE-GR scheme except with two differences.

– Firstly, the user-key generation algorithm is replaced by two algorithms, where one is run by each data user himself/herself called UUserKG, and the other one is run by the AA called AUserKG. The UUserKG algorithm outputs a public and private user-user-key pair. On input a public user-user-key and the master private key of the AA, the AUserKG algorithm outputs a public and private authority-user-key pair and publicly transmits them to the server.
– Secondly, the decryption algorithm is divided into two parts, of which one is run by the server called SDecrypt using the public and private authority user-keys and decryption key, and the other one is run the data user called UDecrypt with the

[8] Please contact the author for the full version.

private user-user-key. The SDecrypt algorithm takes a ciphertext as the input, and outputs a partially decrypted ciphertext. The UDecrypt algorithm takes a partially decrypted ciphertext as the input, and outputs the plaintext.

Assume that for each data user, the server keeps a list of tuples (identity, attributes, public and private authority-user-keys, a set of private attribute-keys), i.e., $(id, \{A_i\}, (pk_{id}, sk_{id}), \{pk_{id}^{Ai}\})$. We detail the concrete construction as follows.

- GSetup. The same as that in the basic ABE-GR construction.
- ASetup. The same as that in the basic ABE-GR construction.
- UUserKG. The data user id randomly chooses $\tau \in Z_p$, and outputs a public and private user-user-key pair $(pk'_{id}, sk'_{id}) = (g^{\tau}, \tau)$.
- AUserKG. The AA randomly chooses $\beta \in Z_p$, and outputs a private and public authority-user-key pair $(sk_{id}, pk_{id}) = ((pk'_{id})^{\alpha}(g^{a})^{\beta}, g^{\beta})$. The AA will publicly send $(sk_{id,}\ pk_{id)}$ to the server.
- PrivKG. The same as that in the basic ABE-GR construction. The AA will publicly send pk_{id}^{Ai} to the server.
- TKeyUp. The same as that in the basic ABE-GR construction. The AA will publicly send $ku_t^{(i)}$ to the server.
- DecKG. The same as that in the basic ABE-GR construction except that it is run by the server rather than the data user.
- Encrypt. The same as that in the basic ABE-GR construction.
- SDecrypt. Given the private authority-user-key and decryption key, the server computes

$$C'_0 = \frac{\hat{e}(C_1, sk_{id}) \prod_{i \in I} \hat{e}(C_4^{(i)}, dk_2^{(i)})}{(\prod_{i \in I} \hat{e}(C_2^{(i)}, pk_{id}) \hat{e}(C_3^{(i)}, dk_1^{(i)}))^{w_i}} = \hat{e}(pk'_{id}, g)^{\alpha\mu},$$

  and sends $\mathrm{CT}' = (id, C_0, C'_0)$ to the data user.
- UDecrypt. The data user computes $M = C_0/(C'_0)^{\frac{1}{\tau}}$ using the private user-user-key.
- Revoke. The same as that in the basic ABE-GR construction.

Remarks. It is worth noticing that the server-aided ABE-GR scheme has an edge over the basic ABE-GR one in both storage and computation overheads. Firstly, each data user in the server-aided ABE-GR construction only needs to keep one short private key, while in the basic ABE-GR one each data user keeps a private key of large size (depending on the size of attribute sets he/she owns and the total number of data users allowed in the system). Secondly, each data user in the server-aided ABE-GR system only needs to perform one exponentiation and no pairing computation to decrypt a ciphertext, while in the basic ABE-GR one each data user needs to perform many exponentiation and pairing computations. Thirdly, there is no secure channel required in the server-aided ABE-GR scheme for private key transmission, but the AA in the basic ABE-GR one needs to send the private user-key and attribute-keys to each data user via a secure channel.

Theorem 2. *Under the decisional q-parallel BDHE assumption, the above server aided ABE-GR scheme is selectively IND-CPA secure.*

Proof. In the proof, it is assumed that if an adversary has issued a private user- user-key query, a private authority-user-key query, and a private attribute-key on attributes $\{A_i\}$ satisfying the challenge access structure $(\mathbb{M}^*, \rho^*)$ on an identity *id*, then at least one attribute in each set of minimum attributes satisfying $(\mathbb{M}^*, \rho^*)$ of this identity *id* is revoked at or before the challenge time period t^*. The proof is similar to that in Theorem 1, and we detail it in the full version of this paper due to the space limit.

4.3 System Analysis

To the best of our knowledge, besides the result in this paper, [1, 5, 20, 25] are also about constructions on revocable ABE from the bilinear maps in the prime-order groups. This paper aims to achieve granular revocation in CP-ABE such that the AA can selectively revoke specific attributes of data users. The KP-ABE scheme with indirect user revocation is proposed in [5] where the AA enables the revocation by disallowing revoked users to update their keys. In [1], a KP-ABE scheme with hybrid user revocation is raised in which a data owner can select to use either direct or indirect revocation mode when encrypting a message. A generic way to build ABE schemes supporting dynamic credentials is elaborated in [20], in which the AA indirectly accomplishes user revocation by preventing revoked data users from updating their keys. In [25], a semi-trusted server is assigned to share the decryption capability with data users such that the server can indirectly revoke a data user by stopping helping this data user with decryption.

Denote "NA" by the meaning of not-applicable. Let R be the number of revoked users, N be the number of all data users, l be the number of attributes presented in the

Table 1. Comparison of properties among revocable ABE (RABE) schemes

	RABE in [5]	RABE in [1]	RABE in [25]	RABE in [20]	Basic ABE-GR	Server-aided ABE-GR
Revocation mode	Indirect	Indirect & direct	Direct	Indirect	Indirect	Indirect
Selective revocation	No	No	No	No	Yes	Yes
Type of ABE	KP-ABE	KP-ABE	CP-ABE	KP-ABE & CP-ABE	CP-ABE	CP-ABE
Key exposure Resistance	No	No	No	No	Yes	Yes
Secure channel	Yes	Yes	Yes	Yes	Yes	No
Server	NA	NA	Semi-trust	NA	NA	Untrust
Size of key updates	$O(R\log(\frac{N}{R}))$	$O(R\log(\frac{N}{R}))$	NA	$O(R\log(\frac{N}{R}))$	$O(mR\cdot\log(\frac{N}{R}))$	$O(mR\cdot\log(\frac{N}{R}))$
Size of key stored by user	$O(l$ log $N)$	$O(l$ log $N)$	$O(1)$	$O(l$ log $N)$ & $O(k$ log $N)$	$O(k$ log $N)$	$O(1)$
Data user's computation overhead	$\geq$ 2(E + P)	$\geq$ 3E + 4P	E	$\geq$ E + P	$\geq$ E + 4P	E

access structure, k be the size of attribute set possed by each data user, and m be the maximum size allowed for k. In Table 1, we compare our revocable systems with the revocable ABE constructions in [1, 5, 20, 25], where "E" and "P" denote the calculation of exponentiation and pairing, respectively. It is straightforward to see that our notion of ABE-GR is the first that achieves selective revocation while preserving desirable properties in terms of both security and efficiency. Additionally, our server-aided ABE-GR scheme greatly reduces the storage and computation overhead incurred to each data user with the help of an untrusted server.

5 Conclusions

In this paper, we introduced a notion called attribute-based encryption with granular revocation (ABE-GR) to achieve selective revocation, where each data user's attributes (or credentials) can be selectively revoked. To our knowledge, there are few works on such a revocation mechanism, and most of the existing revocable ABE schemes aim to revoke a data user from the system such that a revoked data user will become underprivileged to all (newly) encrypted data in the system. Motivated by the key separation technique in distributed ABE [17] where one single AA's workload is split across several AAs and each AA is responsible for at least one specific attribute, we equipped a normal ABE system with a similar technique such that each data user's attribute-keys are composed of key elements (corresponding to different attributes) generated separately but essentially linkable to each other. Thus, each data user's attributes can be selectively revoked by the AA, and a data user can be revoked from the system by separately revoking all of his/her attributes. After the description of security model for SR-ABE, we presented a basic construction of ABE-GR, which utilizes the binary tree data structure to reduce the workload of the AA. Then, we further improved the efficiency by introducing an untrusted server to the proposed ABE-GR scheme to help data users with the workloads incurred in key update and decryption, which we call server-aided ABE-GR. In addition, we formally proved the security of our ABE-GR and server-aided ABE-GR schemes, and compared them with other concrete constructions of revocable ABE that are related to our work.

Acknowledgements. This research work is supported by the Singapore National Research Foundation under the NCR Award Number NRF2014NCR-NCR001-012.

References

1. Attrapadung, N., Imai, H.: Attribute-based encryption supporting direct/indirect revocation modes. In: Parker, M.G. (ed.) IMACC 2009. LNCS, vol. 5921, pp. 278–300. Springer, Heidelberg (2009). doi:10.1007/978-3-642-10868-6_17
2. Attrapadung, N., Imai, H.: Conjunctive broadcast and attribute-based encryption. In: Shacham, H., Waters, B. (eds.) Pairing 2009. LNCS, vol. 5671, pp. 248–265. Springer, Heidelberg (2009). doi:10.1007/978-3-642-03298-1_16

3. Baek, J., Zheng, Y.: Identity-based threshold decryption. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 262–276. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24632-9_19
4. Beimel, A.: Secure schemes for secret sharing and key distribution. Ph.D. thesis, Israel Institute of Technology, June 1996
5. Boldyreva, A., Goyal, V., Kumar, V.: Identity-based encryption with efficient revocation. In: Proceedings of the 2008 ACM Conference on Computer and Communications Security, CCS 2008, Alexandria, Virginia, USA, 27–31 October 2008, pp. 417–426. ACM (2008)
6. Boneh, D., Ding, X., Tsudik, G., Wong, C.: A method for fast revocation of public key certificates and security capabilities. In: 10th USENIX Security Symposium, Washington, D.C., USA, 13–17 August 2001. USENIX (2001)
7. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001). doi:10.1007/3-540-44647-8_13
8. Cui, H., Deng, R.H.: Revocable and decentralized attribute-based encryption. Comput. J. **59** (8), 1220–1235 (2016). doi:10.1093/comjnl/bxw007
9. Ding, X., Tsudik, G.: Simple identity-based cryptography with mediated RSA. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 193–210. Springer, Heidelberg (2003). doi:10.1007/3-540-36563-X_13
10. Hanaoka, Y., Hanaoka, G., Shikata, J., Imai, H.: Identity-based hierarchical strongly key-insulated encryption and its application. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 495–514. Springer, Heidelberg (2005). doi:10.1007/11593447_27
11. Horváth, M.: Attribute-based encryption optimized for cloud computing. In: Italiano, Giuseppe F., Margaria-Steffen, T., Pokorný, J., Quisquater, J.-J., Wattenhofer, R. (eds.) SOFSEM 2015. LNCS, vol. 8939, pp. 566–577. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46078-8_47
12. Lewko, A., Waters, B.: Decentralizing attribute-based encryption. In: Paterson, Kenneth G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 568–588. Springer, Heidelberg (2011). doi:10.1007/978-3-642-20465-4_31
13. Li, J., Li, J., Chen, X., Jia, C., Lou, W.: Identity-based encryption with outsourced revocation in cloud computing. IEEE Trans. Comput. **64**(2), 425–437 (2015)
14. Li, Q., Xiong, H., Zhang, F.: Broadcast revocation scheme in composite-order bilinear group and its application to attribute-based encryption. IJSN **8**(1), 1–12 (2013)
15. Liang, K., Liu, Joseph K., Wong, Duncan S., Susilo, W.: An efficient cloud-based revocable identity-based proxy re-encryption scheme for public clouds data sharing. In: Kutyłowski, M., Vaidya, J. (eds.) ESORICS 2014. LNCS, vol. 8712, pp. 257–272. Springer, Cham (2014). doi:10.1007/978-3-319-11203-9_15
16. Libert, B., Quisquater, J.: Efficient revocation and threshold pairing based cryptosystems. In: Proceedings of the Twenty-Second ACM Symposium on Principles of Distributed Computing, PODC 2003, Boston, Massachusetts, USA, 13–16 July 2003, pp. 163–171. ACM (2003)
17. Müller, S., Katzenbeisser, S., Eckert, C.: Distributed attribute-based encryption. In: Lee, P.J., Cheon, J.H. (eds.) ICISC 2008. LNCS, vol. 5461, pp. 20–36. Springer, Heidelberg (2009). doi:10.1007/978-3-642-00730-9_2
18. Naor, D., Naor, M., Lotspiech, J.: Revocation and tracing schemes for stateless receivers. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 41–62. Springer, Heidelberg (2001). doi:10.1007/3-540-44647-8_3
19. Qin, B., Deng, R.H., Li, Y., Liu, S.: Server-aided revocable identity-based encryption. In: Pernul, G., Ryan, P.Y.A., Weippl, E. (eds.) ESORICS 2015. LNCS, vol. 9326, pp. 286–304. Springer, Cham (2015). doi:10.1007/978-3-319-24174-6_15

20. Sahai, A., Seyalioglu, H., Waters, B.: Dynamic credentials and ciphertext delegation for attribute-based encryption. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 199–217. Springer, Heidelberg (2012). doi:10.1007/978-3-642-32009-5_13
21. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005). doi:10.1007/11426639_27
22. Seo, J.H., Emura, K.: Revocable identity-based encryption revisited: security model and construction. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 216–234. Springer, Heidelberg (2013). doi:10.1007/978-3-642-36362-7_14
23. Wan, Z., Liu, J., Deng, R.H.: HASBE: a hierarchical attribute-based solution for flexible and scalable access control in cloud computing. IEEE Trans. Inf. Forensics Secur. **7**(2), 743–754 (2012)
24. Waters, B.: Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 53–70. Springer, Heidelberg (2011). doi:10.1007/978-3-642-19379-8_4
25. Yang, Y., Ding, X., Lu, H., Wan, Z., Zhou, J.: Achieving revocable fine-grained cryptographic access control over cloud data. In: Desmedt, Y. (ed.) ISC 2013. LNCS, vol. 7807, pp. 293–308. Springer, Cham (2015). doi:10.1007/978-3-319-27659-5_21