doi:10.1093/comjnl/bxw007

Revocable and Decentralized Attribute-Based Encryption

Hui Cui* and Robert H. Deng

Secure Mobile Centre, Singapore Management University, 71 Stamford Road, Singapore 178902, Singapore
**Corresponding author: hcui@smu.edu.sg*

In this paper, we propose a revocable and decentralized attribute-based encryption (ABE) system that splits the task of decryption key generation across multiple attribute authorities (AAs) without requiring any central party such that it achieves attribute revocation by simply stopping updating of the corresponding private key. In our system, a party can easily behave as an AA by creating a public and private key pair without any global communication except the creation for the common system parameters, under which it can periodically issue/update private key components for users that reflect their attributes, and an AA can freely leave the system once its corresponding attribute is revoked without communication with other AAs. In addition, to revoke a user, those AAs that have issued private keys to this user easily cease the key updating process for the user without affecting other AAs' execution. For the construction of our system, the technical barrier is to make private keys collusion resistant. Since in our system each component of a user's private key at a time period may come from different AAs and there is no coordination between these AAs, traditional technique of binding together different components (issued by different AAs) of a private key by randomization cannot be employed. To overcome this, we tie the key components together and prevent collusion attacks between different users by embedding distinct identifiers and a commonly shared time attribute in these components.

Keywords: revocation; decentralization; ABE

Received 9 July 2015; revised 9 December 2015
Handling editor: Yi Mu

1. INTRODUCTION

Public-key encryption can be viewed as a method for a user to share data to a targeted user or device while protecting the confidentiality of the transmitted information from other curious users. This is useful in scenarios where the data provider knows exactly which user it wants to share data with, but this is not realistic in applications where people are identified by their attributes, and the data provider wants to convey data according to some policy based on the privileged user's attributes. Sahai and Waters [1] proposed the notion of attribute-based encryption (ABE) to solve the above problem, where each user is issued an attribute-based private key from an attribute authority (AA) or key generation authority that reflects their attributes, and a data provider can specify access policies to the data over a set of attributes. A user will be able to decrypt a ciphertext if the attributes under its private key satisfy the access policy associated with the ciphertext. One drawback of the original Sahai–Waters approach is that every user must prove its ownership of a set of attributes to a trusted party to obtain the corresponding attribute-based private key for decryption, which implies the existence of a single trusted party who monitors attributes and issues private keys for all users. To overcome this limitation, the concept of multi-authority ABE [2, 3] was introduced, where multiple AAs operate simultaneously, and each distributes private key components corresponding to different sets of attributes. Since a trusted central authority (CA) is still required in the multi-authority ABE schemes in [2, 3], decentralized multi-authority ABE [4–6] was proposed to avert the single security bottleneck incurred by counting on a central AA.

The appearance of ABE affirmatively solves the problem arising in the situations where different users with different attributes are given access to different levels of the encrypted data, but it fails to address the problem that the attributes of a user may change with time. This problem initiated the study of revocation [7] where a periodic key update would only

allow non-revoked users to update their keys to decrypt the newly encrypted data. Considering the ABE setting, there are two main ways to realize revocability: (i) indirect revocation [7, 8] where a sender encrypts with an attribute set and a time attribute, and the AA, with the current revocation list, periodically provides key update information at each time period so that only the non-revoked users can update their keys and thus decrypt the ciphertexts encrypted at the current time period; (ii) direct revocation [8, 9] where a sender specifies the revocation list directly when running the encrypting algorithm (i.e. the sender is required to possess the current revocation list), such that the revocation can be done instantly and does not require the key update phase as in the indirect method. Even though 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, it requires all the data providers to keep the current revocation list, which makes the system be impurely attribute-based, since data providers in the attribute-based setting create ciphertext based only on attributes without caring revocation. On the contrary, indirect revocation does not require the data providers to own the revocation list, but its key update phase is a bottleneck since it requires communication from the AA to all the unrevoked users at all the time periods. In this paper, we focus on the design of efficaciously and indirectly revocable ABE schemes.

Motivated by the fact that decentralization essentially reduces the trust on a central AA, we consider making use of it to alleviate both communication cost and security bottleneck in the single AA setting. That is, we would like the task of a single AA to be split across multiple AAs to reduce the computation overhead. We accomplish this by putting forward a notion of revocable and decentralized ABE with the goal of efficiently achieving revocability in the decentralized ABE systems.

1.1. Our contributions

Based on the scheme given in [6], we present a decentralized ciphertext-policy ABE (CP-ABE) system supporting indirect revocation, which can be regarded as a CP-ABE scheme with two properties: (i) decentralization—any party can become an AA without the requirement of any global coordination except the initialization of the common reference parameters created during the setup phase and (ii) revocability—the AAs periodically update components of attribute-based private keys such that only the non-revoked users can obtain their private keys to decrypt the newly encrypted data in the current time period. Thanks to the decentralization property, in our system, revocation of an attribute, i.e. completely removing an attribute from the system, can be done by simply stopping issuing key update by the corresponding AA. In addition, when an attribute of a user is revoked, the corresponding AA can simply stop updating for this user with no influence on the update for this user from other AAs. In our inial construction, each AA needs to issue each user a new private key at different time slots, which is convenient to be applied in the practice, and we found this methodology under the decentralizing setting can reduce a single AA's workload. However, it would be desirable to reduce the key update size of each AA from linear to logarithmic as in [7]. Therefore, using the binary tree structure [7, 8], we extend our original construction to a new scheme in which the amount of work each AA has to do for key updates is mitigated and the total size of key updates is reduced to logarithmic in the number of users.

In a revocable and decentralized CP-ABE scheme, a party can simply behave as an AA by creating a public and private key pair, with which it can extract and periodically update components of attribute-based private keys for different users. When the attribute of an AA is revoked, this AA can freely terminates its role as an AA by being inactive. Since different AAs operate entirely independently, the failure or corruption of some AAs will not affect the functionality of the uncorrupted AAs. In our system, the concept of global identifier [2, 6] is subtly applied to link together the private key components issued to the same user by multiple AAs, such that any data provider can encrypt data in terms of any time period and any access policy over the attributes issued from any set of AAs.

Our security model for the proposed scheme takes into account all the adversarial capabilities in the standard ABE security notion. First, the adversary should be able to learn the private keys associated with the attribute sets of its choice. Secondly, the adversary should not be able to learn any information about the message encrypted for the challenge set of attributes. In addition, we assume that the adversary is given access to the periodical key updates, which asks the adversary to be unable to learn any information about the message encrypted for any users with attributes containing the time period past the revocation time, i.e. the ciphertext is created after the time of revocation.

Challenges and solutions. Regarding the construction of a revocable and decentralized ABE system, the key challenge is to make it secure against collusion attacks.

Traditionally, to achieve collusion resistance in ABE, the single AA combines different components (corresponding to different attributes) of a user's private key together by randomizing the key. Such randomization makes different key elements for one user compatible with each other, but not work with the components of a key belonging to another user. However, this technique of key randomization does not work in the setting of decentralization, because there is no central party to tie all the pieces together, and each part of a key may come from different AAs who have no coordination and are possibly not aware the existence of each other. To shun this obstacle, we resort to the approach of tying a user's key components together and preventing collusion attacks between different users with different global identities [6]. At a high level, this method, instead of relying on one key generation to bind all the key components together, uses a hash function on the user's global identity GID

to preserve collusion resistance across multiple key generations from different AAs.

On the other hand, when encrypting a message, in our system, a data provider, besides the usual attribute set input $\mathbf{A}$, is also required to specify the current time period Time. In this case, in order to decrypt a ciphertext encrypted for a set of attributes $\mathbf{A}$ and a time period Time, a user must possess the key components for all $i \in \mathbf{A}$ under Time from different AAs which periodically update key components at the start of each time period. Since the time attribute is the same for all users, in case of compromising the security, it is imperative to prevent collusion attacks. This requires that revoked users cannot combine their keys with the update information for the non-revoked users to decrypt a newly created ciphertext which they are not able to decrypt with their old keys. To achieve this, our approach is to bind the time period and the global identifier together during key generation while keeping the global identifier away during the encryption (i.e. the system is still attribute-based).

Our system is proved secure using the recent dual system encryption methodology [10], where the security proof works by first converting the challenge ciphertexts and private keys to a semi-functional form and then arguing the security. Following the variant of the dual system proof technique in [11], we build our scheme using the bilinear groups of composite order on the basis of the decentralization ABE scheme in [6].

1.2. Related work

After Sahai and Waters [1] introduced the notion of ABE, Goyal *et al.* formulated two complimentary forms of ABE [12], CP-ABE [13] and key-policy ABE (KP-ABE) [12]. In a CP-ABE system, the keys are associated with the sets of the attributes and the ciphertexts are associated with the access policies, while 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 sets of the attributes. Since then, various kinds of ABE systems have been proposed [2, 4–6, 8–10, 12, 14–19], including multi-authority ABE [2, 4–6], ABE with full security [10, 17], revocable ABE [7–9, 18] etc.

Multi-authority ABE was proposed by Chase [2] to construct an ABE scheme where the decryption key is created by different AAs. However, the solution presented in [2] need the presence of a single trusted 'CA' to issue each AA a unique key. Chase and Chow [4] tried to remove the CA using a distributed PRF, but the scheme in [4] is achieved under an AND policy and a determined set of AAs. Lin *et al.* [5] gave a threshold-based scheme that is somewhat decentralized, but the set of AAs is fixed ahead of time, and the system is only secure up to collusion of m users. Lewko and Waters [6] presented the first fully decentralized CP-ABE scheme, where any party can become an AA (i.e. the system does not require any CA), and there is no requirement for any global coordination other than the generation of an initial set of common reference parameters.

Boldyreva *et al.* [7] proposed a revocable KP-ABE scheme using an approach of indirect revocation where the AA indirectly enables the revocation by forcing revoked users to be unable to update their keys. In order to eliminate the impedient key update phase in the indirect revocation, Attrapadung and Imai [9] put forth an ABE system with direct revocation, which allows the senders to specify the revocation list directly when encrypting by requiring the senders to keep the current revocation list. Also, Attrapadung and Imai [8] raised a hybrid revocable KP-ABE system under selective security model which allows a sender to select whether to use either direct or indirect revocation mode when encrypting a message. Sahai *et al.* [18] provided the first revocable ABE scheme in a generic way that deals with the problem of efficiently revoking stored data, where the database can periodically update the ciphertexts using only publicly available information stored on the system.

1.3. Organization

The remainder of this paper is organized as follows. In Section 2, we briefly review the definitions associated with this work. In Section 3, after the description of the system framework, we elaborate the security model of revocable and decentralized CP-ABE. In Section 4, we present a specific construction of revocable and decentralized CP-ABE, and analyze its security. In Section 5, we compare our construction with some prior revocable ABE schemes, and then extend our revocable and decentralized ABE system to improve its efficiency. We conclude this paper in Section 6.

2. PRELIMINARIES

In this section, we review some basic cryptographic conceptions and definitions that are related to this work.

2.1. Access structures and linear secret sharing

We review the notions of access structures and linear secret sharing schemes in [6, 16] as follows.

DEFINITION 2.1 (Access Structure). *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* $C \subseteq \mathbb{A}$. *An access structure* (*respectively*, *monotone access structure*) *is a collection* (*respectively*, *monotone collection*) $\mathbb{A}$ *of non-empty subsets of* $\{P_1, \ldots, P_n\}$, *i.e.* $\mathbb{A} \subseteq 2^{\{P_1,\ldots,P_n\}} \setminus \{\emptyset\}$. *The sets in* $\mathbb{A}$ *are called the authorized sets*, *and the sets not in* $\mathbb{A}$ *are called the unauthorized sets.*

In this paper, attributes will play the role of the parties and only the monotone access structures will be considered. Observe that more general access structures [6] can be

(inefficiently) realized with the techniques by letting the negation of an attribute be a separate attribute (this doubles the total number of attributes).

DEFINITION 2.2 (Linear Secret Sharing Schemes (LSSS)). *Let P be a set of parties. Let A be a matrix of size $l \times n$. Let $\rho : \{1, \ldots, l\} \to P$ be a function that maps 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 a matrix A which has l rows and n columns called the share-generating matrix for Π. For $x = 1, \ldots, l$, the xth row of matrix A is labeled by a party $\rho(x)$, where $\rho : \{1, \ldots, l\} \to P$ is a function that maps a row to a party for labeling. Considering that the column vector $\overrightarrow{v} = (s, r_2, \ldots, r_n)$, where $s \in Z_p$ is the secret to be shared and $r_2, \ldots, r_n \in Z_p$ are randomly chosen, then Av is the vector of l shares of the secret s according to Π. The share $(Av)_x$ belongs to party $\rho(x)$.*

It is shown in [6, 16] that every linear secret sharing scheme consistent with the above definition meets the following linear reconstruction property. Suppose that Π is an LSSS for an access structure $\mathbb{A}$. Let $S \in \mathbb{A}$ be an authorized set, and define $I \subseteq \{1, \ldots, l\}$ as $I = \{x \mid \rho(x) \in S\}$. Then the vector $(1, 0, \ldots, 0)$ is in the span of the rows of matrix A indexed by I, and there exist constants $\{w_x \in Z_p\}$ such that if $\{\lambda_x\}$ are valid shares of a secret s according to Π, then $\sum_{x \in I} w_x \lambda_x = s$. According to [6, 16], these constants $\{w_x\}$ can be found in polynomial time in terms of the size of share-generating matrix A.

For the construction under composite order group, the LSSS matrices can be considered over Z_N, where $N = p_1 p_2 p_3$ is a product of three distinct primes. A set S is authorized if the rows of matrix A labeled by elements in S have the vector $(1, 0, \ldots, 0)$ in their span modulo N. Regarding the security proof, it will be further assumed that for an unauthorized set, the corresponding rows of matrix A do not have the vector $(1, 0, \ldots, 0)$ in their span modulo the individual prime factors of N. This is because if an adversary can produce a matrix A over Z_N and an unauthorized set over Z_N that is authorized over Z_{p_i} for $i \in \{1, 2, 3\}$, then this can be used to produce a non-trivial factor of the group order N, which contradicts with the complexity assumptions.

Boolean formulas [6]. Access policies can also be described in terms of monotonic boolean formulas. LSSS access structures are more general, and can be derived from representations as boolean formulas. There are standard techniques to convert any monotonic boolean formula into a corresponding LSSS matrix. The boolean formula can be represented as an access tree, where the interior nodes are AND and OR gates, and the leaf nodes correspond to attributes. The number of rows in the corresponding LSSS matrix will be same as the number of leaf nodes in the access tree.

2.2. Composite-order bilinear groups and complexity assumptions

Let $\mathcal{G}$ be a group generator algorithm which takes a security parameter λ as the input and outputs $(p_1, p_2, p_3, G, G_T, \hat{e})$, where p_1, p_2, p_3 are distinct primes, G and G_T are cyclic groups of order $N = p_1 p_2 p_3$, and $\hat{e} : G \times G \to G_T$ is a map with the following properties [20]:

(1) Bilinear. $\forall g, h \in G, a, b \in Z_N, \hat{e}(g^a, h^b) = \hat{e}(g, h)^{ab}$.
(2) Non-degenerate. $\exists g \in G$ such that $\hat{e}(g, g)$ has order N in G_T.

Assume that the group operations in G and G_T as well as the bilinear map $\hat{e}$ are computable in polynomial time with respect to λ and that the group descriptions of G and G_T include the generators of the respective cyclic groups. Let G_{p_1}, G_{p_2} and G_{p_3} be the subgroups of order p_1, p_2 and p_3 in G, respectively. It is observed in [6] when $h_i \in G_{p_i}$ and $h_j \in G_{p_j}$ for $i \neq j$, $\hat{e}(h_i, h_j)$ is the identity element in G_T. This orthogonality property of G_{p_1}, G_{p_2}, G_{p_3} will be used to implement the semi-functionality in the constructions.

Let $G_{p_1 p_2}$ denote the subgroup of order $p_1 p_2$ in G. Let $g_1 \leftarrow G_{p_1}$ denote that g_1 is chosen to be a random generator of G_{p_1} (so it is not the identity element), and $T_1 \leftarrow G$ denote that T_1 is chosen to be a random generator of G (this is not quite the same as a uniformly random element, but the distributions are negligibly close).

Given a group generator $\mathcal{G}$, the assumptions used in this paper are defined in Fig. 1. The advantage of algorithm $\mathcal{A}$ in breaking Assumption i for $i \in \{1, 2, 3, 4\}$ is defined to be

$$\mathbf{Adv}^{\mathrm{Amp}_i}_{\mathcal{G},\mathcal{A}}(\lambda) = |\Pr[\mathcal{A}(D, T_1) = 1] - \Pr[\mathcal{A}(D, T_2) = 1]|.$$

We say that $\mathcal{G}$ satisfies Assumption i if $\mathbf{Adv}^{\mathrm{Amp}_i}_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of the security parameter λ for any polynomial time algorithm $\mathcal{A}$.

Note that in Assumption 1, T_1 can be written (uniquely) as the product of an element of G_{p_1}, an element of G_{p_2} and an element of G_{p_3}. In this paper, these elements will be referred to as the 'G_{p_1} part of T_1', the 'G_{p_2} part of T_1' and the 'G_{p_3} part of T_1', respectively (Fig. 1).

3. FRAMEWORK AND SECURITY MODEL

In this section, we present the formal definition of revocable and decentralized ABE, after depicting the architecture of our ABE system.

Assumptions	Distributions for $\mathbb{G} = (N = p_1p_2p_3, G, G_T, \hat{e}) \leftarrow \mathcal{G}$
1	$g_1 \leftarrow G_{p_1}$, $D = (\mathbb{G}, g_1)$, $T_1 \leftarrow G, T_2 \leftarrow G_{p_1}$.
2	$g_1, X_1 \leftarrow G_{p_1}, X_2 \leftarrow G_{p_2}, g_3 \leftarrow G_{p_3}$, $D = (\mathbb{G}, g_1, g_3, X_1X_2)$, $T_1 \leftarrow G_{p_1}, T_2 \leftarrow G_{p_1p_2}$.
3	$g_1, X_1 \leftarrow G_{p_1}, Y_2 \leftarrow G_{p_2}, X_3, Y_3 \leftarrow G_{p_3}$, $D = (\mathbb{G}, g_1, X_1X_3, Y_2Y_3)$, $T_1 \leftarrow G_{p_1p_2}, T_2 \leftarrow G_{p_1p_3}$.
4	$g_1 \leftarrow G_{p_1}, g_2 \leftarrow G_{p_2}, g_3 \leftarrow G_{p_3}, a, b, c, d \leftarrow Z_N$ $D = (\mathbb{G}, g_1, g_2, g_3, {g_1}^a, {g_1}^b {g_3}^b, {g_1}^c, {g_1}^{ac} {g_3}^d)$, $T_1 \leftarrow \hat{e}(g_1, g_1)^{abc}, T_2 \leftarrow G_T$.

FIGURE 1. Assumptions 1–4.

3.1. Framework

A revocable and decentralized CP-ABE system is comprised of the following algorithms: global setup algorithm GSetup, AA setup algorithm ASetup, attribute-based private key generation & update algorithm KeyGen & KeyUpdate, encryption algorithm Encrypt and decryption algorithm Decrypt. Assume that the attribute-based private keys of the non-revoked users will be renewed periodically (e.g. every month), and the senders encrypt using the receivers' attributes and the current time period, e.g. 'June of 2015'. Also, there exists a revocation list for each attribute in the system, which stores the identities of all the revoked users associated with the attribute, and is held by the corresponding AA.

(i) GSetup$(1^\lambda) \rightarrow$ GP. Taking the security parameter λ as the input, this global setup algorithm outputs the global parameter GP for the system.

(ii) ASetup(GP) $\rightarrow (apk, ask)$. Taking the global parameter GP as the input, each AA creates its own master public and private key pair (apk, ask).

(iii) KeyGen & KeyUpdate $(\text{GP}, \text{GID}, \text{Time}, rl_i, i, ask) \rightarrow sk_{i,\text{GID}}^{\text{Time}}$. Taking the global parameter GP, a time period Time, a revocation list rl_i, an identity GID, an attribute i belonging to some AA and a master private key ask for this AA as the input, this key generation and update algorithm produces the update key $sk_{i,\text{GID}}^{\text{Time}}$ for this attribute, identity and time tuple.

(iv) Encrypt$(\text{GP}, M, \text{Time}, (A, \rho), \{apk\}) \rightarrow C$. Taking the global parameter GP, a message M, a time period Time, an access matrix (A, ρ) and a set of the master public keys for the relevant AAs as the input, this encryption algorithm outputs a ciphertext C.

(v) Decrypt$(\text{GP}, C, \{sk_{i,\text{GID}}^{\text{Time}}\}) \rightarrow M$. Taking the global parameter GP, the ciphertext C and a collection of update keys corresponding to the attribute, identity and time tuples with the same identity GID and time period Time as the input, this decryption algorithm outputs either the message M when the collection of attributes i satisfies the access matrix corresponding to the ciphertext C, or a symbol $\perp$ indicating the failure of the decryption.

We require that a revocable and decentralized CP-ABE system is correct, meaning that if GP is generated from the global setup algorithm, C is generated from the encryption algorithm on a message M and a time period Time, and $\{sk_{i,\text{GID}}^{\text{Time}}\}$ is a set of private keys obtained from the key generation algorithm for the same identity GID at the time period Time over a set of attributes satisfying the access structure of the ciphertext, then Decrypt$(\text{GP}, C, \{sk_{i,\text{GID}}^{\text{Time}}\}) = M$.

3.2. Security definition

Assuming that the adversary corrupts the AAs statically [2, 4, 6] and make the key generation & update queries adaptively, we define the security for revocable and decentralized CP-ABE encryption systems by the following game between a challenger algorithm $\mathcal{C}$ and an adversary algorithm $\mathcal{A}$.

Let S be the set of all the AAs, and U be the universe of attributes. Suppose that each AA can control multiple attributes, but each attribute is assigned to only one AA. To match the practice, it can be considered as an attribute being the concatenation of an AA's public key and an attribute string, such that if multiple AAs choose the same attribute, these will still correspond to distinct attributes in the system.

(i) Setup. Algorithm $\mathcal{C}$ runs the global setup algorithm. Algorithm $\mathcal{A}$ defines a set of $S' \subseteq S$ of corrupt AAs. For these non-compromised AAs $S - S'$, algorithm $\mathcal{C}$ obtains the public and private key pairs by running the AA setup algorithm, and gives the public keys to algorithm $\mathcal{A}$.

(ii) Phase 1. Algorithm $\mathcal{A}$ makes the key generation & update query to algorithm $\mathcal{C}$. Algorithm $\mathcal{A}$ sends tuples $(i, \text{Time}, \text{GID})$ to algorithm $\mathcal{C}$, where i is an attribute

belonging to an non-corrupted AA, Time is a time period, GID is an identity. Algorithm $\mathcal{C}$ responds by returning the corresponding updated key $sk_{i,\mathrm{GID}}^{\mathrm{Time}}$ to algorithm $\mathcal{A}$.

(iii) Challenge. Algorithm $\mathcal{A}$ chooses two messages M_0 and M_1, a time period Time* and an access matrix (A, ρ) with the constraint that the key generation & update queries $\{sk_{i,\mathrm{GID}}^{\mathrm{Time}^*}\}$ in Phase 1 do not satisfy the access matrix (A, ρ). Let V denote the subset of the rows of A labeled by the attributes controlled by the corrupt AAs. Let V_{GID} denote the subset of the rows of A labeled by attributes i for which algorithm $\mathcal{A}$ has queried $(i, \mathrm{GID}, \mathrm{Time}^*)$ for each global identifier GID. For each GID, to prevent algorithm $\mathcal{A}$ from asking for a set of keys that allow decryption, in combination with any keys that can obtained from the corrupt AAs, we require that the subspace spanned by $V \cup V_{\mathrm{GID}}$ must not include $(1, 0, \ldots, 0)$. In addition, algorithm $\mathcal{A}$ must give algorithm $\mathcal{C}$ the public keys for any corrupt AAs whose attributes appear in the labeling ρ. Algorithm $\mathcal{C}$ chooses a random bit $\beta \in \{0, 1\}$, and sends algorithm $\mathcal{A}$ a challenge ciphertext C^*, an encryption of M_β under access matrix (A, ρ) and time period Time*.

(iv) Phase 2. Algorithm $\mathcal{A}$ continues issuing the key generation & update queries $(i, \mathrm{Time}, \mathrm{GID})$ with the constraint as follows. If algorithm $\mathcal{A}$ makes key generation & update queries $\{sk_{i,\mathrm{GID}}^{\mathrm{Time}^*}\}$, they must not violate the constraint on the challenge matrix (A, ρ) as in the Challenge phase.

(v) Guess. Algorithm $\mathcal{A}$ makes a guess β' for β, and it wins the game if $\beta' = \beta$.

Algorithm $\mathcal{A}$'s advantage in this game is defined as $\Pr[\beta = \beta'] - 1/2$. We say that a revocable and decentralized ABE scheme is secure if all the probabilistic polynomial time (PPT) adversaries have at most a negligible advantage in the security parameter λ.

Remarks. Note that our security definition is a little different from that presented in [8, 18]. First, the definition in [8] considers selective security, where the adversary is required to specify the target time period and attribute set before having access to the public parameter, while ours focuses on full security without such limitation. Secondly, the definition in [18] puts the time in a sequential order, which requires that the adversary, after seeing the target time period, is not allowed to issue key update queries for time period equal to or greater than the target one when the adversary makes private key queries with sufficient attributes to decrypt the target ciphertext, whereas in our model the time period is out of order, and thus the adversary is restricted to make key generation & update queries for time period equal to the target one under the same condition.

4. A REVOCABLE AND ONE-USE MULTI-AUTHORITY CP-ABE SYSTEM

In this section, we describe a revocable and one-use multi-authority CP-ABE scheme, and then prove its security in the random oracle model using the dual system encryption technique [10].

4.1. Construction

The one-use decentralized ABE with revocability system is composed of the following algorithms, of which some are the same as that in [6].

(i) GSetup. The system chooses a composite order bilinear group G of which the group order is a product of three primes as $N = p_1p_2p_3$, a generator g_1 of G_{p_1}, two hash function $H_0 : \{0, 1\}^* \to Z_N$, $H_1 : \{0, 1\}^* \to G$. The global public parameter is $\mathrm{GP} = (N, g_1, H_0, H_1)$.

(ii) ASetup. For every attribute i belonging to the AA (these indices i are not reused between different AAs), the AA randomly chooses $\alpha_i, y_i \in Z_N$ as the private key ask and computes $apk = \hat{e}(g_1, g_1)^{\alpha_i}, {g_1}^{y_i})$ as the public key.

(iii) KeyGen & KeyUpdate. To generate an attributed-based private key for GID $\notin rl_i$ for attribute i at time period Time, the AA computes

$$sk_{i,\mathrm{GID}}^{\mathrm{Time}} = {g_1}^{\alpha_i H_0(\mathrm{Time})} H_1(\mathrm{GID}||\mathrm{Time})^{y_i}.$$

The attribute-based private key for user GID at time period Time is $\{sk_{i,\mathrm{GID}}^{\mathrm{Time}}\}$.

(iv) Encrypt. Let v_x be $A_x \cdot \overrightarrow{v}$, w_x be $A_x \cdot \overrightarrow{w}$, where A_x is row x of access matrix A. It is required that the row labeling ρ of an access matrix to be injective, i.e. each attribute is used at most once. To encrypt a message M under an $n \times l$ access matrix A with ρ mapping the rows to the attributes, the data provider randomly chooses $s \in Z_N$, $\overrightarrow{v} \in Z_N^l$ with s as its first entry, respectively, $\overrightarrow{w} \in Z_N^l$ with 0 as its first entry, $r_x \in Z_N$ for each row A_x of A, and computes

$$\begin{aligned} & C_0 = M \cdot \hat{e}(g_1, g_1)^s, \\ \forall x \quad & C_{1,x} = \hat{e}(g_1, g_1)^{v_x} \hat{e}(g_1, g_1)^{\alpha_{\rho(x)} H_0(\mathrm{Time}) r_x}, \\ & C_{2,x} = {g_1}^{r_x}, \\ & C_{3,x} = {g_1}^{y_{\rho(x)} r_x} {g_1}^{w_x}. \end{aligned}$$

The data provider outputs the ciphertext $C = (C_0, \{C_{1,x}\}, \{C_{2,x}\}, \{C_{3,x}\})$.

(v) Decrypt. Assume that the ciphertext is encrypted under an access matrix (A, ρ). To decrypt a ciphertext C, the receiver first computes $H_1(\mathrm{GID}||\mathrm{Time})$. If the receiver has the private keys $\{sk_{\rho(x),\mathrm{GID}}^{\mathrm{Time}}\}$ for a subset of rows A_x of access matrix A such that $(1, 0, \ldots, 0)$ is in the span of these rows, the receiver proceeds as follows. For each

such x, the receiver computes

$$\frac{C_{1,x} \cdot \hat{e}(H_1(\text{GID}||\text{Time}), C_{3,x})}{\hat{e}(sk^{\text{Time}}_{\rho(x),\text{GID}}, C_{2,x})} = \hat{e}(g_1, g_1)^{v_x} \hat{e}(H_1(\text{GID}||\text{Time}), g_1)^{w_x}.$$

The receiver then chooses $c_x \in Z_N$ such that $\sum_x c_x A_x = (1, 0, \ldots, 0)$, and computes

$$\prod_x (\hat{e}(g_1, g_1)^{v_x} \hat{e}(H_1(\text{GID}||\text{Time}), g_1)^{w_x})^{c_x} = \hat{e}(g_1, g_1)^s, \quad M = \frac{C_0}{\hat{e}(g_1, g_1)^s}.$$

where $v_x = A_x \cdot \overrightarrow{v}$, $w_x = A_x \cdot \overrightarrow{w}$ for $\overrightarrow{v} \cdot (1, 0, \ldots, 0) = s$, $\overrightarrow{w} \cdot (1, 0, \ldots, 0) = 0$,

Multi-use multi-authority CP-ABE. The construction we presented will be proved secure under the restriction that attributes are used only once. It has been showed in [6] how to construct a secure multi-use multi-authority CP-ABE system where attributes can be used multiple times in an access matrix from a secure multi-authority CP-ABE scheme where attributes are used only once with a simple encoding technique. Since our revocable and decentralized CP-ABE scheme is built upon the scheme in [6], this encoding approach can be applied to our construction as well.

Removing secure channels. In our system, each available AA (whose attribute has not been revoked) needs to update users' private keys at a regular basis. This means that all the users have to regularly contact each AA, prove their attributes and get new private keys, which implies that all the available AAs must be online for all such communications, and a secure channel must be established between each AA and each user to transmit the private key. This becomes extremely annoying for a very large number of users. In order to avoid the need for frequent interaction and a secure channel, we can ask each AA to encrypt the new keys of non-revoked users under their global identities and the previous time period [7], and forward the ciphertexts to these users. With this approach, for every non-revoked user, each AA is required to perform one key generation and one encryption per key update.

4.2. Security

In order to reduce the security of our revocable and decentralized attribute-based encryption system, the dual system encryption technique [10] is applied, in which the keys and the ciphertexts are divided as normal and semi-functional: the normal keys can decrypt the semi-functional ciphertexts, the semi-functional keys can decrypt the normal ciphertexts, but the semi-functional keys cannot decrypt the semi-functional ciphertexts. The proof is proceeded over a sequence of games, where firstly the challenge ciphertext is changed to be semi-functional, and then the keys are changed to be semi-functional one by one. These games are indistinguishable from each other, which can be proved from the fact that the simulator itself cannot tell the form of the key being turned from normal to semi-functional by decrypting a semi-functional ciphertext.

Since the construction of our system is very similar to that in [6], we begin with the definitions of the semi-functional ciphertexts and keys based on that in [6], which are not used in the real system but will be used in the proof. The semi-functional ciphertexts will include elements from subgroups G_{p_2} and G_{p_3}, and the semi-functional keys will be Types 1 and 2, containing elements in G_{p_2} and G_{p_3}, respectively. When a semi-functional key of Type 1 is used to decrypt a semi-functional ciphertext, the extra elements from G_{p_2} in the key will be tied with the extra G_{p_2} elements in the ciphertext, which will cause the decryption to fail. When a semi-functional key of Type 2 is used to decrypt a semi-functional ciphertext, the extra elements from G_{p_3} in the key will be tied with the extra G_{p_3} elements in the ciphertext, which will cause the failure of the decryption. Specifically, each attribute i is assigned with two random elements $z_i, t_i \in Z_N$, and they will be common to the semi-functional ciphertexts and keys. Note that z_i, t_i are fixed for each attribute i, and will not be changed for different users.

(i) *Semi-functional ciphertexts.* To create a semi-functional ciphertext, the encryption algorithm is run to obtain a normal ciphertext $(C'_0, C'_{1,x}, C'_{2,x}, C'_{3,x} \forall x)$. Let g_2, g_3 be the generators of G_{p_2} and G_{p_3}, respectively. Let $\overrightarrow{u_2}$, $\overrightarrow{u_3} \in Z_N^l$ be two randomly chooses vectors. For each row A_x of the access matrix A, define $\delta_x = A_x \cdot \overrightarrow{u_2}$, $\sigma_x = A_x \cdot \overrightarrow{u_3}$. Let B be the subset of the rows of the access matrix A whose corresponding attributes belong to the corrupt AAs. Let $\bar{B}$ be the subset of the rows of the access matrix A whose corresponding attributes are controlled by the uncorrupt AAs. Thus, the semi-functional ciphertext is formed as $C_0 = C'_0$,

$$\begin{aligned}
\forall x, A_x \in \bar{B} \quad & C_{1,x} = C'_{1,x}, \\
& C_{2,x} = C'_{2,x} g_2{}^{\gamma_x} g_3{}^{\psi_x}, \\
& C_{3,x} = C'_{3,x} g_2{}^{\delta_x + \gamma_x z_{\rho(x)}} g_3{}^{\sigma_x + \psi_x t_{\rho(x)}}, \\
\forall x, A_x \in B \quad & C_{1,x} = C'_{1,x}, \quad C_{2,x} = C'_{2,x}, \\
& C_{3,x} = C'_{3,x} g_2{}^{\delta_x} g_3{}^{\sigma_x},
\end{aligned}$$

where γ_x, $\psi_x \in Z_N$ are randomly chosen from Z_N. A ciphertext is said to be notionally semi-functional when all the values δ_x are the shares of 0.

(ii) *Semi-functional keys.* The attribute-based private key for identity GID and time period Time is defined to be the collection of $H_1(\text{GID}||\text{Time})$ and all the keys $sk^{\text{Time}}_{i,\text{GID}}$ for attributes i belonging to the uncorrupt AAs queried by the adversary throughout the game. The semi-functional keys for an identity GID and a time period Time will be of Type 1 or Type 2.

Let $H_1'(\text{GID}||\text{Time})$ be a random element of G_{p_1}. To create a semi-functional key of Type 1, the random oracle's outputs on GID||Time is defined to be

$$H_1(\text{GID}||\text{Time}) = H_1'(\text{GID}||\text{Time}){g_2}^{c_1},$$

where $c_1 \in Z_N$. Thus, a semi-functional key $sk_{i,\text{GID}}^{\text{Time}}$ for an attribute i controlled by an uncorrupt AA can be generated by producing a normal key $sk_{i,\text{GID}}^{\text{Time}'}$ and then setting

$$sk_{i,\text{GID}}^{\text{Time}} = sk_{i,\text{GID}}^{\text{Time}'} {g_2}^{c_1 z_i}.$$

To create a semi-functional key of Type 2, the random oracle's outputs on GID||Time is defined to be

$$H_1(\text{GID}||\text{Time}) = H_1'(\text{GID}||\text{Time}){g_3}^{c_1},$$

where $c_1 \in Z_N$. Thus, a semi-functional key $sk_{i,\text{GID}}^{\text{Time}}$ for an attribute i belonging to an uncorrupt AA can be generated by producing a normal key $sk_{i,\text{GID}}^{\text{Time}'}$ and then setting

$$sk_{i,\text{GID}}^{\text{Time}} = sk_{i,\text{GID}}^{\text{Time}'} {g_3}^{c_1 t_i}.$$

Note that when a semi-functional key of Type 1 is used to decrypt a semi-functional ciphertext, the additional elements $\hat{e}(g_2, g_2)^{c_1 \delta_x}$ prevent the decryption from succeeding, except when the values δ_x are shares of 0 (i.e. it is a notionally semi-functional ciphertext). Likewise, when a semi-functional key of Type 2 is used to decrypt a semi-functional ciphertext, the additional elements $\hat{e}(g_3, g_3)^{c_1 \delta_x}$ make the decryption fail.

Taking the one-use restriction on attributes throughout the proof into consideration, which means that the row labeling ρ of the challenge ciphertext access matrix (A, ρ) must be injective, we formally define the sequence of the games on the basis of that in [6].

The first game, $\text{Game}_{\text{Real}}$, is the real security game. The next game, $\text{Game}_{\text{Real}'}$, is similar to the real security game, except that the random oracle maps identities GID and time periods Time to the random elements of G_{p_1} instead of G.

Game_0 is similar to $\text{Game}_{\text{Real}'}$, except that the ciphertext given to the adversary is semi-functional. Let q be the number of identities GID and time periods Time for which the adversary makes the key queries $sk_{i,\text{GID}}^{\text{Time}}$. $\text{Game}_{j,1}$ and $\text{Game}_{j,2}$ for $j \in [1, q]$ are defined as follows:

(i) $\text{Game}_{j,1}$. This is similar to Game_0, except that for the first $j-1$ queried pairs of identity and time period, the received keys are semi-functional of Type 2, and the received key for the jth queried pair of identity and time period is semi-functional of Type 1. The remaining keys are normal.

(ii) $\text{Game}_{j,2}$. This is similar to Game_0, except that for the first j queried pairs of identity and time period, the received keys are semi-functional of Type 2. The remaining keys are normal. Note that in $\text{Game}_{q,2}$, all the keys are semi-functional of Type 2.

(iii) $\text{Game}_{\text{Final}}$. In this game, all the attribute-based private keys are semi-functional of Type 2, and the ciphertext is a semi-functional encryption of a random message. Note that the adversary has advantage 0 in this game.

These games are indistinguishable according to the following lemmas. Recall that the proof of security is considered under the restriction that each attribute is used at most one time in the access matrix, i.e. ρ is injective. Note that some contents of the proofs exactly follow that in [6].

LEMMA 4.1. *Assuming that there exists a polynomial time algorithm $\mathcal{A}$ such that $\boldsymbol{Adv}_{\mathcal{A}}^{\text{Game}_{\text{Real}}} - \boldsymbol{Adv}_{\mathcal{A}}^{\text{Game}_{\text{Real}'}} = \epsilon$, then there exists construct a polynomial time algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption* 1.

Proof. Given N, g_1, T, algorithm $\mathcal{B}$ chooses a hash function $H_0 : \{0,1\}^* \to Z_N$, and will simulate either $\text{Game}_{\text{Real}}$ or $\text{Game}_{\text{Real}'}$ with algorithm $\mathcal{A}$, depending on the value of T. Algorithm $\mathcal{B}$ outputs g_1 as the public generator of G_{p_1} and N as the group order. Algorithm $\mathcal{A}$ specifies a set $S' \subseteq S$ of the corrupt AAs for S being the set of all the AAs in the system. For each attribute i belonging to an uncorrupt AA, algorithm $\mathcal{B}$ randomly chooses $\alpha_i, y_i \in Z_N$, and sends algorithm $\mathcal{A}$ the public parameter $\hat{e}(g_1, g_1)^{\alpha_i}$, ${g_1}^{y_i}$.

For the queries to the random oracle for $H_1(\text{GID}||\text{Time})$ issued by algorithm $\mathcal{A}$, algorithm $\mathcal{B}$ randomly chooses $h_{\text{GID,Time}} \in Z_N$ and sets

$$H_1(\text{GID}||\text{Time}) = T^{h_{\text{GID,Time}}}.$$

Algorithm $\mathcal{B}$ stores this value so that it can respond consistently when $H_1(\text{GID}||\text{Time})$ is queried again. If T is a generator of G_{p_1}, this value will be a random element of G_{p_1}. If T is a generator of G, this value will be a random element of G.

For the queries (GID, Time, i) to the key generation & update oracle from algorithm $\mathcal{A}$, algorithm $\mathcal{B}$ extracts the corresponding private key by using the key generation & update algorithm with the known values α_i and y_i. Note that if GID||Time has not been issued to the random oracle yet, algorithm $\mathcal{B}$ sets $H_1(\text{GID}||\text{Time})$ as above.

When it comes to the challenge phase, algorithm $\mathcal{A}$ gives algorithm $\mathcal{B}$ two messages M_0, M_1, a time period Time* and an access matrix (A, ρ), as well as the public parameter for the corrupted AAs appearing in the matrix. To create the challenge ciphertext for M_β where $\beta \in \{0,1\}$ is a random bit chosen by algorithm $\mathcal{B}$, algorithm $\mathcal{B}$ runs the encryption algorithm.

If T is a generator of G, then algorithm $\mathcal{B}$ has correctly simulated $\text{Game}_{\text{Real}}$. If T is a generator of G_{p_1}, then algorithm $\mathcal{B}$ has correctly simulated $\text{Game}_{\text{Real}'}$. Thus, algorithm $\mathcal{B}$ can use algorithm $\mathcal{A}$ to attain advantage ϵ in breaking Assumption 1.

This completes the proof of Lemma 4.1. □

LEMMA 4.2. *Assuming that there exists a polynomial time algorithm $\mathcal{A}$ such that $\boldsymbol{Adv}_{\mathcal{A}}^{\text{Game}_{\text{Real}'}} - \boldsymbol{Adv}_{\mathcal{A}}^{\text{Game}_0} = \epsilon$, then there*

exists a polynomial time algorithm $\mathcal{B}$ with advantage negligibly close to ϵ in breaking Assumption 1.

Proof. Given N, g_1, T, algorithm $\mathcal{B}$ chooses a hash function $H_0 : \{0,1\}^* \rightarrow Z_N$, and will simulate either $\text{Game}_{\text{Real}'}$ or Game_0 with algorithm $\mathcal{A}$, depending on the value of T. Algorithm $\mathcal{B}$ outputs g_1 as the public generator of G_{p_1} and N as the group order. Algorithm $\mathcal{A}$ specifies a set $S' \subseteq S$ of the corrupt AAs for S being the set of all the AAs in the system. For each attribute i belonging to an uncorrupt AA, algorithm $\mathcal{B}$ randomly chooses $\alpha_i, y_i \in Z_N$, and gives algorithm $\mathcal{A}$ the public parameter $\hat{e}(g_1, g_1)^{\alpha_i}$, ${g_1}^{y_i}$.

For the queries to the random oracle for $H_1(\text{GID}||\text{Time})$ from algorithm $\mathcal{A}$, algorithm $\mathcal{B}$ randomly chooses $h_{\text{GID,Time}} \in Z_N$, and sets

$$H_1(\text{GID}||\text{Time}) = g_1^{h_{\text{GID,Time}}}.$$

Algorithm $\mathcal{B}$ stores this value so that it can respond consistently when $H_1(\text{GID}||\text{Time})$ is queried again.

For the queries (GID, Time, i) to the key generation & update oracle from algorithm $\mathcal{A}$, algorithm $\mathcal{B}$ extracts the corresponding private key by running the key generation & update algorithm with α_i and y_i which are known to algorithm $\mathcal{B}$. Again, if GID||Time has not been queried to the random oracle yet, algorithm $\mathcal{B}$ sets $H_1(\text{GID}||\text{Time})$ as above.

For the challenge phase, algorithm $\mathcal{A}$ gives algorithm $\mathcal{B}$ two messages M_0, M_1, a time period Time* and an access matrix (A, ρ), as well as the public parameter g^{y_i}, $\hat{e}(g_1, g_1)^{\alpha_i}$ for attributes i belonging to the corrupt AAs that are included in the access matrix (A, ρ). Algorithm $\mathcal{B}$ chooses a random bit $\beta \in \{0,1\}$, and encrypts M_β as follows. First, algorithm $\mathcal{B}$ randomly chooses $s \in Z_N$, and sets $C_0 = M_\beta \cdot \hat{e}(g_1, g_1)^s$. Algorithm $\mathcal{B}$ then chooses two vectors, $\overrightarrow{v} = (s, v_2, \ldots, v_l)$, $\overrightarrow{w} = (0, w_2, \ldots, w_l)$, where $v_2, \ldots, v_l$, $w_2, \ldots, w_l$ are randomly chosen from Z_N.

Let $v_x = A_x \cdot \overrightarrow{v}$ and $w_x = A_x \cdot \overrightarrow{w}$. Let B denote the subset of the rows of A whose corresponding attributes belong to the corrupted AAs. Let $\bar{B}$ be the subset of the rows of A whose corresponding attributes belong to the uncorrupt AAs. For each row $A_x \in B$, algorithm $\mathcal{B}$ randomly chooses $r_x \in Z_N$, and computes the ciphertext as

$$\begin{aligned}
C_{1,x} &= \hat{e}(g_1, g_1)^{v_x} (\hat{e}({g_1}^{H_0(\text{Time}^*)}, g_1)^{\alpha_{\rho(x)}})^{r_x}, \\
C_{2,x} &= {g_1}^{r_x}, \\
C_{3,x} &= ({g_1}^{y_{\rho(x)}})^{r_x} T^{w_x}.
\end{aligned}$$

For each row $A_x \in \bar{B}$, algorithm $\mathcal{B}$ randomly chooses $r'_x \in Z_N$, implicitly sets $r_x = r r'_x$ such that ${g_1}^r$ is the G_{p_1} part of T (T will be embedded into the calculation with shares w_x), and computes the ciphertext as

$$\begin{aligned}
C_{1,x} &= \hat{e}(g_1, g_1)^{v_x} (\hat{e}({g_1}^{H_0(\text{Time}^*)}, T)^{\alpha_{\rho(x)} r'_x}), \\
C_{2,x} &= T^{r'_x}, \\
C_{3,x} &= T^{y_{\rho(x)} r'_x} T^{w_x}.
\end{aligned}$$

Note that the G_{p_1} part of T^{w_x} is ${g_1}^{A_x \cdot r\overrightarrow{w}}$, where $r\overrightarrow{w}$ is a random vector with the first coordinate equal to 0. Thus, if $T \in G_{p_1}$, this is a correctly distributed normal ciphertext. If $T \in G$, then this is a semi-functional ciphertext with $\delta_x = A_x \cdot c\overrightarrow{w}$ modulo p_2 where ${g_2}^c$ is the G_{p_2} part of T, $\sigma_x = A_x \cdot d\overrightarrow{w}$ modulo p_3 where ${g_3}^d$ is the G_{p_3} part of T, ${g_2}^{\gamma_x}$ equals the G_{p_2} part of $T^{r'_x}$, ${g_3}^{\psi_x}$ equals the G_{p_3} part of $T^{r'_x}$, $z_{\rho(x)} = y_{\rho(x)}$ modulo p_2, and $t_{\rho(x)} = y_{\rho(x)}$ modulo p_3.

This is correctly computed. Since r'_x, $y_{\rho(x)}$ are randomly chosen in Z_N, their values modulo p_1, modulo p_2 and modulo p_3 are uncorrelated by the Chinese Remainder Theorem [6]. This means that γ_x, ψ_x, $z_{\rho(x)}$, $t_{\rho(x)}$ are randomly distributed, $w_2, \ldots, w_l$ of $\overrightarrow{w}$ are also randomly distributed modulo p_2, p_3, but both δ_x and σ_x are the shares of 0 from algorithm $\mathcal{B}$'s point of view. Hence, it remains to be argued that these appear to be the shares of a random exponent in the view of algorithm $\mathcal{A}$.

Note that the shares δ_x, σ_x for the rows $A_x \in B$ are information-theoretically revealed to algorithm $\mathcal{A}$, but the space R spanned by these rows cannot include the vector $(1, 0, \ldots, 0)$ (assume that this holds modulo p_2). This means that there exists a vector $\overrightarrow{u}$ such that $\overrightarrow{u}$ is orthogonal to R modulo p_2, but $\overrightarrow{u}$ is not orthogonal to $(1, 0, \ldots, 0)$. Fix a basis including the vector $\overrightarrow{u}$, and write $c\overrightarrow{w} = \overrightarrow{w'} + a\overrightarrow{u}$ for some a modulo p_2 and $\overrightarrow{w'}$ in the span of the other basis elements. Note that $\overrightarrow{w}$ is uniformly distributed in this space (modulo p_2) and reveals no information about a (modulo p_2). Now, the first coordinate of $c\overrightarrow{w}$ modulo p_2 depends on the value of a, and the shares δ_x for $A_x \in B$ contain no information about a (since $\overrightarrow{u}$ is orthogonal to R). The only information algorithm $\mathcal{A}$ receives about the value of a appears in the exponents of the form $\delta_x + \gamma_x z_{\rho(x)}$, where the $z_{\rho(x)}$ is a new random value each time that appears nowhere else (recall that ρ is constrained to be injective). As long as γ_x does not equal 0 modulo p_2 ($\gamma_x = 0$ with only negligible probability), this means that any value of δ_x can be explained by $z_{\rho(x)}$ taking on a particular value. Since $z_{\rho(x)}$ is uniformly random, this means that no information about the value of a is revealed. Hence, the value being shared is information-theoretically hidden, and the shares δ_x (and similarly σ_x) are well distributed in algorithm $\mathcal{A}$'s view.

Thus, when $T \in G_{p_1}$, algorithm $\mathcal{B}$ correctly simulates $\text{Game}_{\text{Real}'}$. When $T \in G$, algorithm $\mathcal{B}$ properly simulates Game_0 with probability negligibly close to 1. Therefore, algorithm $\mathcal{B}$ can use algorithm $\mathcal{A}$ to obtain advantage negligibly close to ϵ in breaking Assumption 1.

This completes the proof of Lemma 4.2. □

LEMMA 4.3. *Assuming that there exists a polynomial time algorithm $\mathcal{A}$ such that $\boldsymbol{Adv}_{\mathcal{A}}^{\text{Game}_{j-1,2}} - \boldsymbol{Adv}_{\mathcal{A}}^{\text{Game}_{j,1}} = \epsilon$, then there exists construct a polynomial time algorithm $\mathcal{B}$ with advantage negligibly close to ϵ in breaking Assumption* 2.

Proof. Given g_1, g_3, X_1X_2, T, algorithm $\mathcal{B}$ chooses a hash function $H_0 : \{0,1\}^* \rightarrow Z_N$, and will simulate either $\text{Game}_{j-1,2}$ or

$\text{Game}_{j,1}$ with algorithm $\mathcal{A}$, depending on the value of T. Algorithm $\mathcal{B}$ outputs g_1 as the public generator of G_{p_1} and N as the group order. Algorithm $\mathcal{A}$ specifies a set $S' \subseteq S$ of the corrupt AAs for S being the set of all the AAs in the system. For each attribute i belonging to an uncorrupt AA, algorithm $\mathcal{B}$ randomly chooses $\alpha_i, y_i \in Z_N$, and gives algorithm $\mathcal{A}$ the public parameter $\hat{e}(g_1, g_1)^{\alpha_i}$, ${g_1}^{y_i}$.

Let GID_k, Time_k denote the kth identity and time period queried by algorithm $\mathcal{A}$. For the queries issued to the random oracle for $H_1(\text{GID}_k||\text{Time}_k)$ from algorithm $\mathcal{A}$, algorithm $\mathcal{B}$ responds as follows:

(i) If $k > j$, then algorithm $\mathcal{B}$ randomly chooses $h_{\text{GID}_k,\text{Time}_k} \in Z_N$, and sets

$$H_1(\text{GID}_k||\text{Time}_k) = {g_1}^{h_{\text{GID}_k,\text{Time}_k}}.$$

(ii) If $k < j$, then algorithm $\mathcal{B}$ randomly chooses $h_{\text{GID}_k,\text{Time}_k} \in Z_N$, and sets

$$H_1(\text{GID}_k||\text{Time}_k) = (g_1 g_3)^{h_{\text{GID}_k,\text{Time}_k}}.$$

Note that these are random elements of $G_{p_1 p_3}$ since the values of $h_{\text{GID}_k,\text{Time}_k}$ modulo p_1 and modulo p_3 are uncorrelated).

(iii) If $k = j$, then algorithm $\mathcal{B}$ randomly chooses $h_{\text{GID}_j,\text{Time}_j} \in Z_N$, and then sets

$$H_1(\text{GID}_j||\text{Time}_j) = T^{h_{\text{GID}_j,\text{Time}_j}}.$$

In all the three cases, algorithm $\mathcal{B}$ stores the corresponding values in order to respond consistently when $H_1(\text{GID}_k||\text{Time}_k)$ is queried again.

For the queries $(i, \text{GID}_k, \text{Time}_k)$ to the key generation & update oracle from algorithm $\mathcal{A}$, algorithm $\mathcal{B}$ responds as follows. If $H_1(\text{GID}_k||\text{Time}_k)$ has already been fixed, then algorithm $\mathcal{B}$ turns to the stored value. Otherwise, algorithm $\mathcal{B}$ creates $H_1(\text{GID}_k||\text{Time}_k)$ according to k as above. Algorithm $\mathcal{B}$ forms the key as

$$sk_{i,\text{GID}_k}^{\text{Time}_k} = {g_1}^{H_0(\text{Time}_k)\alpha_i} H_1(\text{GID}_k||\text{Time}_k)^{y_i}.$$

Notice that when $k < j$, algorithm $\mathcal{B}$ generates the semi-functional keys of Type 2 with t_i being congruent to y_i modulo p_3 (these are uncorrelated from the values of y_i modulo p_1 which appear in the public parameter). Also, recall that the values t_i are fixed for every attribute, and do not vary across different keys. When $k > j$, algorithm $\mathcal{B}$ generates the normal keys. When $k = j$, algorithm $\mathcal{B}$ generates a normal key if $T \in G_{p_1}$ and a semi-functional key of Type 1 if $T \in G_{p_1 p_2}$.

For the challenge phase, algorithm $\mathcal{A}$ gives algorithm $\mathcal{B}$ two messages M_0, M_1, a time period Time^* and an access matrix (A, ρ). In addition, algorithm $\mathcal{A}$ sends algorithm $\mathcal{B}$ the public parameter g^{y_i}, $\hat{e}(g_1, g_1)^{\alpha_i}$ for attributes i belonging to the corrupt AAs which are included in the access matrix (A, ρ). In order to generate a challenge ciphertext, algorithm $\mathcal{B}$ chooses a random bit $\beta \in \{0, 1\}$, and encrypts M_β as follows. Note that algorithm $\mathcal{B}$ will output a notionally semi-functional ciphertext, but this will be hidden from the perspective of algorithm $\mathcal{A}$. First, algorithm $\mathcal{B}$ randomly chooses $s \in Z_N$, and sets $C_0 = M_\beta \cdot e(g_1, g_1)^s$. Then algorithm $\mathcal{B}$ chooses three vectors, $\overrightarrow{v} = (s, v_2, \ldots, v_l)$, $\overrightarrow{w} = (0, w_2, \ldots, w_l)$ and $\overrightarrow{u} = (u_1, \ldots, u_l)$, where $v_2, \ldots, v_l$, $w_2, \ldots, w_l$, $u_1, \ldots, u_l$ are randomly chosen from Z_N.

Let $v_x = A_x \cdot \overrightarrow{v}$, $w_x = A_x \cdot \overrightarrow{w}$, and $\sigma_x = A_x \cdot \overrightarrow{u}$. Let algorithm B be the subset of the rows of A whose corresponding attributes belong to the corrupted AAs. Let $\bar{B}$ be the subset of the rows of A whose corresponding attributes belong to the uncorrupt AAs. For each row $A_x \in B$, algorithm $\mathcal{B}$ randomly chooses $r_x \in Z_N$, and computes the ciphertext as

$$\begin{aligned} C_{1,x} &= \hat{e}(g_1, g_1)^{v_x} (\hat{e}(g_1, g_1)^{\alpha_{\rho(x)}})^{H_0(\text{Time}^*) r_x}, \\ C_{2,x} &= {g_1}^{r_x}, \\ C_{3,x} &= ({g_1}^{y_{\rho(x)}})^{r_x} (X_1 X_2)^{w_x} {g_3}^{\sigma_x}. \end{aligned}$$

For each row $A_x \in \bar{B}$, algorithm $\mathcal{B}$ randomly chooses $\psi_x, r'_x \in Z_N$, implicitly sets $r_x = r r'_x$ such that ${g_1}^r$ is X_1, and computes the ciphertext as

$$\begin{aligned} C_{1,x} &= \hat{e}(g_1, g_1)^{v_x} \hat{e}(g_1, X_1 X_2)^{\alpha_{\rho(x)} H_0(\text{Time}^*) r'_x}, \\ C_{2,x} &= X_1 {X_2}^{r'_x} {g_3}^{\psi_x}, \\ C_{3,x} &= X_1 {X_2}^{y_{\rho(x)} r'_x} {g_3}^{y_{\rho(x)} \psi_x} (X_1 X_2)^{w_x} {g_3}^{\sigma_x}. \end{aligned}$$

Note that ${X_1}^{w_x}$ is ${g_1}^{A_x \cdot r\overrightarrow{w}}$, and $r\overrightarrow{w}$ is a random vector with first coordinate equal to 0. This is a semi-functional ciphertext with $\delta_x = A_x \cdot c\overrightarrow{w}$ modulo p_2 where ${g_2}^c$ is X_2, ${g_2}^{\gamma_x}$ equals ${X_2}^{r'_x}$, $z_{\rho(x)} = y_{\rho(x)}$ modulo p_2, and $t_{\rho(x)} = y_{\rho(x)}$ modulo p_3.

This is a correctly generated. Note that since r'_x, $y_{\rho(x)}$ are randomly chosen in Z_N, their values modulo p_1 and modulo p_2 are uncorrelated. This means that γ_x, ψ_x, $z_{\rho(x)}$, $t_{\rho(x)}$ are randomly distributed. It is clear that σ_x is properly distributed, since it is a share of a random vector. The entries $w_2, \ldots, w_l$ of $\overrightarrow{w}$ are also randomly distributed modulo p_2, but the values of δ_x are the shares of 0 from the perspective of algorithm $\mathcal{B}$. It remains to be argued that these appear to be the shares of a random exponent in the view of algorithm $\mathcal{A}$.

Let the space R denote the span of the rows of A whose attributes are in B and the rows whose attributes $\rho(x)$ are queried by algorithm $\mathcal{A}$ with identity and time period GID_j, Time_j. This space cannot include the vector $(1, 0, \ldots, 0)$ (assume that this modulo p_2), so there is some vector $\overrightarrow{u'}$ which is orthogonal to R modulo p_2 and not orthogonal to $(1, 0, \ldots, 0)$. Write $c\overrightarrow{w} = \overrightarrow{w'} + a\overrightarrow{u'}$ for some a modulo p_2 and $\overrightarrow{w'}$ in the span of the other basis vectors. Note that $\overrightarrow{w'}$ is uniformly distributed in this space, and reveals no information about a. The value of the first coordinate of $c\overrightarrow{w}$ modulo p_2 depends on the value of a, but the shares δ_x for $A_x \in B$ contain no information

about a. The information algorithm $\mathcal{A}$ receives about the value of a emerges in the exponents of the form $\delta_x + \gamma_x z_{\rho(x)}$, where $z_{\rho(x)}$ is a new random value each time that has not appeared anywhere else (recall that ρ is constrained to be injective). Note that these $z_{\rho(x)}$ values modulo p_2 do not occur in any keys for identities and time periods not equal to $\mathrm{GID}_j, \mathrm{Time}_j$, since these keys are either normal or semi-functional of Type 2, they do not have components in G_{p_2}. As long as γ_x does not equal 0 ($\gamma_x = 0$ with only negligible probability), this means that any value of δ_x can be explained by $z_{\rho(x)}$ taking on a particular value. Since $z_{\rho(x)}$ is uniformly random, this means that no information about the value of a modulo p_2 is revealed. Hence, the value being shared is information-theoretically hidden, and the values of δ_x are properly distributed in the view of algorithm $\mathcal{A}$.

Though it is hidden from algorithm $\mathcal{A}$, the fact that δ_x can only be made to be the shares of 0 is crucial here (i.e. algorithm $\mathcal{B}$ can only make a notionally semi-functional ciphertext). If algorithm $\mathcal{B}$ attempts to test the semi-functionality of the jth key for itself by making a ciphertext the key could decrypt, decryption would succeed regardless of the presence of G_{p_2} components, since the δ_x's are shares of 0. Hence algorithm $\mathcal{B}$ would not be able to tell whether the jth key is semi-functional of Type 1 or normal.

In summary, when $T \in G_{p_1}$, algorithm $\mathcal{B}$ properly simulates $\mathrm{Game}_{j-1,2}$. When $T \in G_{p_1 p_2}$, algorithm $\mathcal{B}$ properly simulates $\mathrm{Game}_{j,1}$ with probability negligibly close to 1. Hence, algorithm $\mathcal{B}$ can use algorithm $\mathcal{A}$ to obtain advantage negligibly close to ϵ in breaking Assumption 2.

This completes the proof of Lemma 4.3. □

LEMMA 4.4. *Assuming that there exists a polynomial time algorithm $\mathcal{A}$ such that $\boldsymbol{Adv}_{\mathcal{A}}^{\mathrm{Game}_{j,1}} - \boldsymbol{Adv}_{\mathcal{A}}^{\mathrm{Game}_{j,2}} = \epsilon$, then there exists a polynomial time algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption* 3.

Proof. Given $N, g_1, X_1X_3, Y_2Y_3, T$, algorithm $\mathcal{B}$ chooses a hash function $H_0 : \{0,1\}^* \to Z_N$, and will simulate either $\mathrm{Game}_{j,1}$ or $\mathrm{Game}_{j,2}$ with algorithm $\mathcal{A}$, depending on the value of T. Algorithm $\mathcal{B}$ outputs g_1 as the public generator of G_{p_1} and N as the group order. Algorithm $\mathcal{A}$ specifies a set $S' \subseteq S$ of the corrupt AAs for S being the set of all the AAs in the system. For each attribute i belonging to an uncorrupt AA, algorithm $\mathcal{B}$ chooses random exponents $\alpha_i, y_i \in Z_N$, and gives algorithm $\mathcal{A}$ the public parameter $\hat{e}(g_1, g_1)^{\alpha_i}, {g_1}^{y_i}$.

Let $\mathrm{GID}_k, \mathrm{Time}_k$ denote the kth identity and time period queried by algorithm $\mathcal{A}$. For the queries to the random oracle for $H_1(\mathrm{GID}_k||\mathrm{Time}_k)$ from algorithm $\mathcal{A}$, respectively, algorithm $\mathcal{B}$ responds as follows:

(i) If $k > j$, then algorithm $\mathcal{B}$ randomly chooses $h_{\mathrm{GID}_k,\mathrm{Time}_k} \in Z_N$, and sets

$$H_1(\mathrm{GID}_k||\mathrm{Time}_k) = {g_1}^{h_{\mathrm{GID}_k,\mathrm{Time}_k}}.$$

(ii) If $k < j$, then algorithm $\mathcal{B}$ randomly chooses $h_{\mathrm{GID}_k,\mathrm{Time}_k} \in Z_N$, and sets

$$H_1(\mathrm{GID}_k||\mathrm{Time}_k) = (X_1X_3)^{h_{\mathrm{GID}_k,\mathrm{Time}_k}}.$$

Note that these are the random elements of $G_{p_1p_3}$ since the values of $h_{\mathrm{GID}_k,\mathrm{Time}_k}$ and h_{Time_k} modulo p_1 and modulo p_3 are uncorrelated).

(iii) If $k = j$, then algorithm $\mathcal{B}$ randomly chooses $h_{\mathrm{GID}_k,\mathrm{Time}_k} \in Z_N$, and sets

$$H_1(\mathrm{GID}_k||\mathrm{Time}_k) = T^{h_{\mathrm{GID}_k,\mathrm{Time}_k}}.$$

In all the three cases, algorithm $\mathcal{B}$ stores the corresponding values to respond consistently when $H_1(\mathrm{GID}_k||\mathrm{Time}_k)$ is queried again.

For the queries $(i, \mathrm{GID}_k, \mathrm{Time}_k)$ to the key generation & update oracle from algorithm $\mathcal{A}$, algorithm $\mathcal{B}$ responds as follows. If $H_1(\mathrm{GID}_k||\mathrm{Time}_k)$ has already been fixed, then algorithm $\mathcal{B}$ uses the stored value. Otherwise, algorithm $\mathcal{B}$ creates $H_1(\mathrm{GID}_k||\mathrm{Time}_k)$ according to k as above. Algorithm $\mathcal{B}$ forms the key as

$$sk_{i,\mathrm{GID}_k}^{\mathrm{Time}_k} = g_1^{H_0(\mathrm{Time}_k)\alpha_i} H_1(\mathrm{GID}_k||\mathrm{Time}_k)^{y_i}.$$

Note that when $k < j$, algorithm $\mathcal{B}$ simulates the semi-functional keys of Type 2 for t_i being congruent to y_i modulo p_3 (these are uncorrelated from the values of y_i modulo p_1 which appear in the public parameter). When $k > j$, algorithm $\mathcal{B}$ simulates the normal keys. When $k = j$, algorithm $\mathcal{B}$ simulates a semi-functional key of Type 1 if $T \in G_{p_1p_2}$ and a semi-functional key of Type 2 if $T \in G_{p_1p_3}$.

For the challenge phase, algorithm $\mathcal{A}$ gives algorithm $\mathcal{A}$ two messages M_0, M_1, a time period Time^* and an access matrix (A, ρ). Additionally, algorithm $\mathcal{A}$ sends algorithm $\mathcal{B}$ the public parameter g^{y_i}, $\hat{e}(g_1, g_1)^{\alpha_i}$ for attributes i belonging to the corrupt AAs which are included in the access matrix (A, ρ). Algorithm $\mathcal{B}$ chooses a random bit $\beta \in \{0,1\}$, and encrypts M_β as follows to generate a challenge ciphertext. First, algorithm $\mathcal{B}$ randomly chooses $s \in Z_N$, and sets $C_0 = M_\beta \cdot \hat{e}(g_1, g_1)^s$. Then, algorithm $\mathcal{B}$ chooses three vectors, $\overrightarrow{v} = (s, v_2, \ldots, v_l)$, $\overrightarrow{w} = (0, w_2, \ldots, w_l)$, $\overrightarrow{u} = (u_1, \ldots, u_l)$, where $v_2, \ldots, v_l$, $w_2, \ldots, w_l, u_2, \ldots, u_l$ are randomly chosen from Z_N.

Let $x = A_x \cdot \overrightarrow{v}$, $w_x = A_x \cdot \overrightarrow{w}$ and $\delta_x = A_x \cdot \overrightarrow{u}$. Let B denote the subset of the rows of A whose corresponding attributes belong to the corrupt AAs. Let $\bar{B}$ be the subset of the rows of A whose corresponding attributes belong to the uncorrupt AAs. For each row A_x, algorithm $\mathcal{B}$ randomly chooses $r_x \in Z_N$. If row $A_x \in B$, the ciphertext is computed as

$$C_{1,x} = \hat{e}(g_1, g_1)^{v_x} (\hat{e}(g_1, g_1)^{\alpha_{\rho(x)}})^{H_0(\mathrm{Time}^*) r_x},$$
$$C_{2,x} = {g_1}^{r_x},$$
$$C_{3,x} = ({g_1}^{y_{\rho(x)}})^{r_x} (g_1)^{w_x} (Y_2Y_3)^{\delta_x}.$$

If row $A_x \in \bar{B}$, the ciphertext is computed as

$$C_{1,x} = \hat{e}(g_1, g_1)^{v_x} \hat{e}(g_1, g_1)^{\alpha_{\rho(x)} H_0(\text{Time}^*) r_x},$$
$$C_{2,x} = {g_1}^{r_x} (Y_2 Y_3)^{r_x},$$
$$C_{3,x} = {g_1}^{y_{\rho(x)} r_x} {g_1}^{w_x} (Y_2 Y_3)^{y_{\rho(x)} r_x} (Y_2 Y_3)^{\delta_x}.$$

Note that this implicitly sets $z_{\rho(x)} \equiv y_{\rho(x)}$ modulo p_2 and $t_{\rho(x)} \equiv y_{\rho(x)}$ modulo p_3. Since these values are uncorrelated, and the sharing vector $\overrightarrow{u}$ is random modulo p_2 and p_3, this is a correctly created semi-functional ciphertext.

Thus, when $T \in G_{p_1 p_2}$, algorithm $\mathcal{B}$ successfully simulates $\text{Game}_{j,1}$. When $T \in G_{p_1 p_3}$, algorithm $\mathcal{B}$ successfully simulates $\text{Game}_{j,2}$. Hence, algorithm $\mathcal{B}$ can use algorithm $\mathcal{A}$ to obtain advantage ϵ in breaking Assumption 3.

This completes the proof of Lemma 4.4. □

LEMMA 4.5. *Assuming that there exists a polynomial time algorithm $\mathcal{A}$ such that $\boldsymbol{Adv}_{\mathcal{A}}^{\text{Game}_{q,2}} - \boldsymbol{Adv}_{\mathcal{A}}^{\text{Game}_{\text{Final}}} = \epsilon$, then there exists a polynomial time algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption* 4.

Proof. Given $g_1, g_2, g_3, {g_1}^a, {g_1}^b {g_3}^b, {g_1}^c, {g_1}^{ac} {g_3}^d, T$, algorithm $\mathcal{B}$ chooses a hash function $H_0 : \{0,1\}^* \to Z_N$, and will simulate either $\text{Game}_{q,2}$ or $\text{Game}_{\text{Final}}$ with algorithm $\mathcal{A}$, depending on the value of T. Algorithm $\mathcal{B}$ outputs g_1 as the public generator of G_{p_1} and N as the group order. Algorithm $\mathcal{A}$ specifies a set $S' \subseteq S$ of the corrupt AAs for S being the set of all the AAs in the system. For each attribute i belonging to an uncorrupt AA, algorithm $\mathcal{B}$ randomly chooses $\alpha_i', y_i' \in Z_N$, and gives algorithm $\mathcal{A}$ the public parameter

$$\hat{e}(g_1, g_1)^{\alpha_i} = \hat{e}({g_1}^a, {g_1}^b {g_3}^b) \hat{e}(g_1, g_1)^{\alpha_i'},$$
$$g_1^{y_i} = {g_1}^a {g_1}^{y_i'}.$$

Note that this sets $\alpha_i = ab + \alpha_i'$, $y_i = a + y_i'$.

For the queries the random oracle for $H_1(\text{GID}||\text{Time})$ from algorithm $\mathcal{A}$, algorithm $\mathcal{B}$ randomly chooses $f_{\text{GID,Time}}$, $h_{\text{GID,Time}} \in Z_N$, and sets

$$H_1(\text{GID}||\text{Time}) = (g_1^b g_3^b)^{-H_0(\text{Time})} g_1^{f_{\text{GID,Time}}} g_3^{h_{\text{GID,Time}}}.$$

Algorithm $\mathcal{B}$ stores this values.

For the queries $(i, \text{Time}, \text{GID})$ to the key generation & update oracle from algorithm $\mathcal{A}$, algorithm $\mathcal{B}$ responds as follows. If both $H_1(\text{GID}||\text{Time})$ has already been fixed, algorithm $\mathcal{B}$ spots the stored value. Otherwise, algorithm $\mathcal{B}$ creates $H_1(\text{GID}||\text{Time})$ as above. Considering only the subgroup 1 part, algorithm $\mathcal{B}$ needs to compute

$$\begin{aligned} sk_{i,\text{GID}}^{\text{Time}} &= g_1^{(ab+\alpha_i') H_0(\text{Time})} ({g_1}^{-bH_0(\text{Time}) + f_{\text{GID,Time}}})^{a+y_i'} \\ &= g_1^{H_0(\text{Time}) ab + H_0(\text{Time})\alpha_i'} {g_1}^{-baH_0(\text{Time}) + f_{\text{GID,Time}} a} \\ &\quad \times {g_1}^{-bH_0(\text{Time}) y_i' + y_i' f_{\text{GID,Time}}} \\ &= (g_1^a)^{f_{\text{GID,Time}}} g_1^{H_0(\text{Time})\alpha_i' + f_{\text{GID,Time}} y_i'} {g_1}^{-by_i' H_0(\text{Time})}. \end{aligned}$$

Note that only ${g_1}^{-by_i' f_{\text{Time}}}$ in the above formula is unknown to algorithm $\mathcal{B}$, and the rest can either be easily computed or be cancelled. To have ${g_1}^{-by_i' f_{\text{Time}}}$, algorithm $\mathcal{B}$ computes $(g_1^b g_3^b)^{-y_i' f_{\text{Time}}}$, and thus the full key can be formed as

$$(g_1^a)^{f_{\text{GID,Time}}} g_1^{H_0(\text{Time})\alpha_i' + f_{\text{GID,Time}} y_i'}$$
$$(g_1^b g_3^b)^{-y_i' H_0(\text{Time})} g_3^{h_{\text{GID,Time}} y_i'}.$$

For the challenge phase, algorithm $\mathcal{A}$ gives algorithm $\mathcal{B}$ two messages M_0, M_1, a time period Time^* and an access matrix (A, ρ). Also, algorithm $\mathcal{A}$ sends algorithm $\mathcal{B}$ the public parameter g^{y_i}, $\hat{e}(g_1, g_1)^{\alpha_i}$ for attributes i belonging to the corrupt AAs which are included in the access matrix (A, ρ). Let B denote the subset of the rows of A whose corresponding attributes belong to the corrupt AAs. Let $\bar{B}$ be the subset of the rows of A whose corresponding attributes belong to the uncorrupt AAs. Algorithm $\mathcal{B}$ chooses a random bit $\beta \in \{0,1\}$, and encrypts M_β as follows.

Algorithm $\mathcal{B}$ computes $C_0 = M_\beta \cdot T$ with the setting $s = abc$. If $T = \hat{e}(g_1, g_1)^{abc}$, this will be an encryption of M_β. If T is random, this will be an encryption of a random message.

Algorithm $\mathcal{B}$ chooses a random vector $\overrightarrow{u_1} \in Z_N^l$, of which the first entry is 1 and $\overrightarrow{u_1}$ is orthogonal to all the rows in B (such a vector is assumed to be of existence, since otherwise the access matrix is illegal or a non-trivial factor of N can be found, violating the complexity assumptions). Algorithm $\mathcal{B}$ also chooses a vector $\overrightarrow{u_2} \in Z_N^l$, of which the first entry is 0 and the rest are randomly chosen. Define the vector $\overrightarrow{v} = abc\overrightarrow{u_1} + \overrightarrow{u_2}$ (note that this vector is uniformly random from algorithm $\mathcal{A}$'s view). Let $v_x = A_x \cdot \overrightarrow{v} = abcA_x \cdot \overrightarrow{u_1} + A_x \cdot \overrightarrow{u_2}$. Since algorithm $\mathcal{B}$ is not able to compute $\hat{e}(g_1, g_1)^{abcA_x \cdot \overrightarrow{u_1}}$ for the rows $A_x \in \bar{B}$, it sets

$$r_x = -\frac{cA_x \cdot \overrightarrow{u_1}}{H_0(\text{Time}^*)} + r_x',$$

where r_x' is randomly chosen from Z_N. Then

$$\begin{aligned} &v_x + H_0(\text{Time}^*)\alpha_{\rho(x)} r_x \\ &\quad = abcA_x \cdot \overrightarrow{u_1} + A_x \cdot \overrightarrow{u_2} \\ &\qquad + H_0(\text{Time}^*)(ab + \alpha_{\rho(x)}') \left(-\frac{cA_x \cdot \overrightarrow{u_1}}{H_0(\text{Time}^*)} + r_x' \right) \\ &\quad = A_x \cdot \overrightarrow{u_2} - c\alpha_{\rho(x)}' A_x \cdot \overrightarrow{u_1} \\ &\qquad + H_0(\text{Time}^*)abr_x' + H_0(\text{Time}^*)\alpha_{\rho(x)}' r_x'. \end{aligned}$$

Thus, algorithm $\mathcal{B}$ can compute $C_{1,x}$ for $A_x \in \bar{B}$ as

$$\begin{aligned} C_{1,x} = \hat{e}(g_1, {g_1}^c)^{-\alpha_{\rho(x)}' A_x \cdot \overrightarrow{u_1}} \hat{e}({g_1}^a, {g_1}^b {g_3}^b)^{H_0(\text{Time}^*) r_x'} \\ \times \hat{e}(g_1, g_1)^{A_x \cdot \overrightarrow{u_2} + H_0(\text{Time}^*)\alpha_{\rho(x)}' r_x'}. \end{aligned}$$

For the rows $A_x \in B$, algorithm $\mathcal{B}$ randomly chooses $r_x \in Z_N$, and computes

$$C_{1,x} = \hat{e}(g_1, g_1)^{A_x \cdot \overrightarrow{u_2}} (\hat{e}(g_1, g_1)^{\alpha_{\rho(x)}})^{H_0(\text{Time}^*) r_x}.$$

Note that $v_x = A_x \cdot \overrightarrow{u_2}$ for these rows because $\overrightarrow{u_1}$ is orthogonal to A_x. For the rows $A_x \in \bar{B}$, algorithm $\mathcal{B}$ randomly chooses $\gamma_x \in Z_N$, and computes $C_{2,x}$ as

$$C_{2,x} = (g_1{}^c)^{-A_x \cdot \overrightarrow{u_1} + r'_x} (g_2 g_3)^{\gamma_x}.$$

Note that the values of γ_x modulo p_2 and p_3 are uncorrelated, so this is correctly generated. For the rows $A_x \in B$, algorithm $\mathcal{B}$ can simply compute $C_{2,x} = g_1{}^{r_x}$.

Algorithm $\mathcal{B}$ now randomly chooses a vector $\overrightarrow{w}$ with the first entry equal to 0 and the other entries randomly chosen from Z_N, and a random vector $\overrightarrow{u_3}$ whose entries are all randomly chosen from Z_N. Let $w_x = A_x \cdot \overrightarrow{w}$ and $\delta_x = A_x \cdot \overrightarrow{u_3}$. For the rows $A_x \in B$, since

$$\begin{aligned} y_{\rho(x)} r_x &= (a + y'_{\rho(x)})(-cA_x \cdot \overrightarrow{u_1} + r'_x) \\ &= -acA_x \cdot \overrightarrow{u_1} - cy'_{\rho(x)} A_x \cdot \overrightarrow{u_1} + r'_x a + y'_{\rho(x)} r'_x, \end{aligned}$$

algorithm $\mathcal{B}$ computes $C_{3,x}$ as

$$\begin{aligned} C_{3,x} = {} & g_1{}^{w_x} (g_1{}^c)^{-y'_{\rho(x)} A_x \cdot \overrightarrow{u_1}} (g_1{}^a)^{r'_x} g_1{}^{y'_{\rho(x)} r'_x} \\ & \times (g_1{}^{ac} g_3{}^d)^{-A_x \cdot \overrightarrow{u_1}} (g_2 g_3)^{\delta_x + \gamma_x y'_{\rho(x)}}. \end{aligned}$$

This is consistent with $t_{\rho(x)}$ being congruent to $y'_{\rho(x)}$ modulo p_3 in the keys. Note that the sharing vector in subgroups G_{p_2} and G_{p_3} is $-d\overrightarrow{u_1} + \overrightarrow{u_3}$, which is random modulo p_2 and modulo p_3. For the rows $A_x \in B$, algorithm $\mathcal{B}$ computes $C_{3,x}$ as

$$C_{3,x} = (g_1{}^{y_{\rho(x)}})^{r_x} g_1{}^{w_x} (g_2 g_3)^{A_x \cdot \overrightarrow{u_3}}.$$

The sharing vector is consistent here because $\overrightarrow{u_1}$ is orthogonal to all of these rows A_x. This is a correctly generated semi-functional ciphertext with $s = abc$. If $T = \hat{e}(g_1, g_1)^{abc}$, this is a semi-functional encryption of M_β, and algorithm $\mathcal{B}$ has simulated $\text{Game}_{q,2}$. If T is random, this is a semi-functional encryption of a random message, and algorithm $\mathcal{B}$ has simulated $\text{Game}_{\text{Final}}$. Hence, algorithm $\mathcal{B}$ can use algorithm $\mathcal{A}$ to obtain advantage ϵ in breaking Assumption 4.

This completes the proof of Lemma 4.5. □

5. DISCUSSION

In this section, we make a comparison among some revocable ABE schemes, and then show some interesting extensions to our revocable and decentralized ABE system.

5.1. Comparisons

As far as we know, besides the work in this paper, Refs [7, 8, 18] are also about ABE systems supporting indirect revocation. These systems manage to solve revocation problem from different point of views. This paper aims to achieve indirect revocation in a decentralized CP-ABE system such that the AAs can indirectly accomplish revocation by stopping updating the keys for the revoked users. In [7], a revocable KP-ABE scheme is proposed using indirect revocation where the AA enables the revocation by forcing the revoked users to be unable to update their keys. In [8], a hybrid revocable KP-ABE system is raised under selective security model which allows a sender to select whether to use either direct or indirect revocation mode when encrypting a message. In [18], the first revocable ABE scheme is provided in a generic way that deals with the problem of efficiently revoking stored data, where the database can periodically update the ciphertexts using only publicly available information stored on the system. Compared with other ABE schemes supporting indirect revocation, the major advantage of our system is that there are no prefixed values on the numbers of users, revoked users, attributes and time periods allowed for the system.

In Table 1, we compare the scalability of our system with those KP-ABE or CP-ABE schemes under indirect revocation in [7, 8, 18], in which 'preset' means that the value is confirmed during the setup phase of the system. Let m be the maximum number of attributes allowed for the system, t be the maximum number of time periods allowed for the system, d be the maximum number of attributes allowed to be revoked for the system. From Table 1, it is not difficult to see that our construction has an edge in achieving scalability among indirectly revocable ABE schemes, because an attribute can be freely added or revoked at any time without affecting the execution of the system.

In Table 2, we compare the efficiency of our system with the revocable ABE schemes in [7, 8, 18] and the basic scheme of our system [6], where the values corresponding to the first two and the next three schemes show the number of group elements in G_p and G_N, respectively, and the symbol—denotes not-applicable. Denote by $|sk|$, $|ct|$, $|uk|$, $|pk|$ the size of user

TABLE 1. Comparison of scalability among revocable ABE systems.

	KP in [7]	KP in [8]	KP in [18]	CP in [18]	Our CP
Preset m	√	√	√	√	×
Preset d	√	√	√	√	×
Preset t	√	√	√	√	×

TABLE 2. Comparison of efficiency among ABE systems. Note that below we assume that each AA in our system is in charge of one attribute.

	$\|sk\|$	$\|ct\|$	$\|uk\|$	$\|pk\|$
KP in [7]	$2l\log(n)+1$	$k+2$	$2\left(r\log\left(\frac{n}{r}\right)+r\right)$	$m+3$
KP in [8]	$2(l+1)\log(n)$	$k+3$	$2\left(r\log\left(\frac{n}{r}\right)+r\right)$	$\gg m+3$
KP in [18]	$2ln$	$k+2$	–	$m+4$
CP in [18]	$2(k+2)n$	$2(l+1)$	–	$m+4$
CP in [6]	n	$3l+1$	–	3
Our CP	n	$3l+1$	$n-r$	3

private key, ciphertext overhead, user update key and fixed system public key, respectively. Let r be the number of the revoked users, n be the number of all the users, l be the size of the number of the attributes presented in the access structure, k be the size of the attribute set, m be the maximum size allowed for k. According to Table 2, we can see that our system performs better in alleviating AA's computation consumption than other revocable systems when $l > n/2$, $r > n/5$, yet accomplishes the same efficiency of the original decentralized ABE scheme without revocation in [6].

5.2. Extension to logarithmic size of key updates

In our previous scheme, each AA works linearly in the number of users, so it could be desirable to find countermeasures to ameliorate it. We can simply employ to the above scheme the complete-subtree method [21] to reduce the size of key updates from linear to logarithmic in the number of users [7, 8, 18].

To begin with, we recall the node selection algorithm KUNodes introduced in [7], which computes the minimal set of nodes for which key update needs to be published so that only non-revoked users at each time period are able to decrypt the ciphertexts. In our new scheme, each AA will be assigned a binary tree BT_i. Let BT_i be a binary tree with N leaves corresponding to N users. Denote $\mathbf{root}_i$ by the root node of the tree BT_i. If L is a leaf node then $\mathrm{Path}(L)$ denotes the set of nodes on the path from L to $\mathbf{root}_i$, including both L and $\mathbf{root}_i$. If L is a non-leaf node, then L_l, L_r denote left and right child of L. The algorithm KUNodes takes the binary tree BT_i, the revocation list rl_i and a revocation time period Time as input, it marks all the ancestors of revoked nodes as revoked, and then outputs all the non-revoked children of revoked nodes. Refer to [7] for a pictorial depiction and details of the formal definition about KUNodes.

$\mathrm{KUNodes}(\mathrm{BT}_i, rl_i, \mathrm{Time})$
$X, Y \leftarrow \emptyset$.
$\forall (L_i, \mathrm{Time}_i) \in rl_i$,
  if $\mathrm{Time}_i \leq \mathrm{Time}$, then add $\mathrm{Path}(L_i)$ to X.
$\forall \tau \in X$,
  if $\tau_l \notin X$, then add τ_l to Y;
  if $\tau_r \notin X$, then add τ_r to Y.
If $Y = \emptyset$, then add $\mathbf{root}_i$ to Y.
Return Y.

Below we give the algorithms of our new construction, which mostly follow that in the previous scheme except that the KeyGen & KeyUpdate algorithm in the prior construction will be replaced by three algorithms.

(i) GSetup. The same as in the previous construction.
(ii) ASetup. The same as in the previous construction.
(iii) Private key generation. This algorithm chooses an unassigned leaf note L from BT and store GID in this node. For each node $\tau \in \mathrm{Path}(L)$, if it is undefined, it chooses $a_{i,\tau} \in Z_N$, stores $a_{i,\tau}$ in the node τ. Otherwise, it fetches the stored $a_{i,\tau}$, and computes

$$P_{i,\tau} = g_1^{a_{i,\tau}+\alpha_i} \cdot H_1(\mathrm{GID})^{y_i}.$$

It outputs $\{\tau, P_{i,\tau}\}_{\tau \in \mathrm{Path}(L)}$ and the update state st_i as BT_i.
(iv) Key update generation. For each node $\tau \in \mathrm{KUNodes}(\mathrm{BT}_i, rl_i, \mathrm{Time})$, this algorithm chooses $a_{i,\tau} \in Z_N$ and stores $a_{i,\tau}$ in the node τ if it is undefined. Otherwise, it fetches the stored $a_{i,\tau}$ and computes

$$Q_{i,\tau} = g_1^{a_{i,\tau} H_0(\mathrm{Time})+\alpha_i} \cdot H_1(\mathrm{Time})^{s_{i,\tau}},$$
$$Q'_{i,\tau} = g_1^{s_{i,\tau}},$$

where $s_{i,\tau}$ is randomly chosen from Z_N. It outputs $\{\tau, Q_{i,\tau}, Q'_{i,\tau}\}_{\tau \in \mathrm{KUNode}(\mathrm{BT}_i, rl_i, \mathrm{Time})}$.
(v) Decryption key generation. This algorithm takes $\{j, P_{i,j}\}_{\tau \in J}$, $\{j', Q_{i,j'}, Q'_{i,j'}\}_{\tau \in J'}$ for some set of nodes J, J' as input. If $J \cap J' = \emptyset$, it outputs $\perp$. Otherwise, it chooses an arbitrary $\tau \in (J \cap J')$, and outputs

$$sk_{i,\mathrm{GID}}^{\mathrm{Time}} = (\tau, P_{i,\tau}, Q_{i,\tau}, Q'_{i,\tau}).$$

(vi) Encrypt. This algorithm follows that in the previous construction except that the ciphertext $C = (C_0, \{C_{1,x}, C_{2,x}, C_{3,x}, C_{4,x}\})$ as

$$\begin{aligned} & C_0 = M \cdot \hat{e}(g_1, g_1)^s, \\ \forall x \quad & C_{1,x} = \hat{e}(g_1, g_1)^{v_x} \hat{e}(g_1, g_1)^{\alpha_{\rho(x)} r_x}, \\ & C_{2,x} = g_1^{r_x}, \quad C_{4,x} = H_1(\mathrm{Time})^{r_x}, \\ & C_{3,x} = g_1^{y_{\rho(x)} r_x} g_1^{w_x}. \end{aligned}$$

(vii) Decrypt. This algorithm follows that in the previous construction except that it firstly computes

$$K_{0,x} = \frac{\hat{e}(H_1(\mathrm{GID}), C_{3,x})}{\hat{e}(P_{x,\tau}, C_{2,x})} = \frac{\hat{e}(H_1(\mathrm{GID}), g_1)^{w_x}}{\hat{e}(g_1^{a_{x,\tau}+\alpha_x}, g_1^{r_x})},$$
$$K_{1,x} = \frac{\hat{e}(Q'_{x,\tau}, C_{4,x})}{\hat{e}(Q_{x,\tau}, C_{2,x})} = \frac{1}{\hat{e}(g_1^{a_{x,\tau} H_0(\mathrm{Time})+\alpha_x}, g_1^{r_x})}.$$

Then it computes

$$\begin{aligned} & C_{1,x} \cdot K_{0,x}^{\frac{H_0(\mathrm{Time})}{H_0(\mathrm{Time})-1}} \cdot K_{1,x}^{\frac{1}{1-H_0(\mathrm{Time})}} \\ & = \hat{e}(g_1, g_1)^{v_x} \cdot (\hat{e}(H_1(\mathrm{GID}), g_1)^{w_x})^{\frac{H_0(\mathrm{Time})}{H_0(\mathrm{Time})-1}} \\ & = \hat{e}(g_1, g_1)^{v_x} \cdot \hat{e}(H_1(\mathrm{GID})^{\frac{H_0(\mathrm{Time})}{H_0(\mathrm{Time})-1}}, g_1)^{w_x}. \end{aligned}$$

Thereafter, it outputs the encrypted message as that in the previous construction.

To prove the security, the proving technique in [7] needs to be combined to the proof of our previous construction. We omit the details here.

5.3. Improvements

Our system can be improved as follows.

(i) *Prime order groups*. It would be more pleasant to use groups of prime order which can potentially result in more efficient systems (via faster group operations). One approach is to replace the group order of three primes in our scheme with a group order of one prime, but as far as we know, the security of such a construction could not be reduced under a non-interactive assumption [6]. Also, there is a class of general transformations of converting pairing-based cryptosystems from composite-order groups to prime-order groups that has been discussed in [22], yet none of them is suitable to our construction.

(ii) *Removing the random oracle*. The random oracle model has been proved to be extremely useful for designing simple, efficient and highly practical solutions for many problems, but from a theoretical point of view, a security proof in the random oracle model is only a heuristic indication of the system's security when instantiated with a particular hash function [23]. As mentioned in [12], the method adopted to obtain large universe constructions for ABE is applicable to remove the random oracle, but it is far from ideal [12] as this makes the system vulnerable to collusion attacks.

In the future, we would like to improve our system by proposing more feasible and compatible solutions to these problems.

6. CONCLUSIONS

Inspired by the observation that multi-authority can reduce the trust on a single AA, we put forth a notion of revocable and decentralized ABE to alleviate the AA's work in the periodical update phase where the revoked users will not be provided with the key update information, which splits the exclusive AA's role across multiple AAs. In our system, any party can become an AA without any global communication except the generation for the public parameters. A party can simply act as an AA by creating a public and private key pair, and then issuing and periodically updating the private keys for different users that reflect their attributes under its master private key and the public parameter. Besides, if an AA's attribute is revoked, this AA can freely leave the system by being unavailable, which has no impact on other AAs. When some attributes of a user is revoked, the corresponding AAs can simply stop updating for this user without affecting other AAs' function. A data provider can encrypt data over any time period and any boolean formula over the attributes issued from any chosen set of AAs. In addition, there is no requirement for any central AA in our system.

In building such a system, the major technical crux is to make it secure against collusion attacks. Because in our system each component of a user's private key will come from different AAs and there is no coordination between these AAs, the key randomizing technique of binding together different components (representing different AAs) of the private key cannot be employed. To overcome this, we trickly tie the key components together and prevent collusion attacks between different users with distinct global identifiers and the same time attribute.

FUNDING

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

REFERENCES

[1] Sahai, A. and Waters, B. (2005) Fuzzy Identity-based Encryption. *Proc. 24th Annual Int. Conf. Theory and Applications of Cryptographic Techniques, Advances in Cryptology – EUROCRYPT 2005*, Aarhus, Denmark, May 22–26, Lecture Notes in Computer Science 3494, pp. 457–473. Springer.

[2] Chase, M. (2007) Multi-authority Attribute based Encryption. *Proc. 4th Theory of Cryptography Conf., TCC 2007*, Amsterdam, The Netherlands, February 21–24, Lecture Notes in Computer Science 4392, pp. 515–534. Springer.

[3] Müller, S., Katzenbeisser, S. and Eckert, C. (2008) Distributed Attribute-based Encryption. *11th Int. Conf. Information Security and Cryptology – ICISC 2008, Revised Selected Papers*, Seoul, Korea, December 3–5, Lecture Notes in Computer Science 5461, pp. 20–36. Springer.

[4] Chase, M. and Chow, S.S.M. (2009) Improving Privacy and Security in Multi-authority Attribute-based Encryption. *Proc. 2009 ACM Conf. Computer and Communications Security, CCS 2009*, Chicago, IL, USA, November 9–13, pp. 121–130. ACM.

[5] Lin, H., Cao, Z., Liang, X. and Shao, J. (2010) Secure threshold multi authority attribute based encryption without a central authority. *Inf. Sci.*, **180**, 2618–2632.

[6] Lewko, A.B. and Waters, B. (2011) Decentralizing attribute-based encryption. *Proc. 30th Annual Int. Conf. Theory and Applications of Cryptographic Techniques, Advances in Cryptology – EUROCRYPT 2011*, Tallinn, Estonia, May 15–19, Lecture Notes in Computer Science 6632, pp. 568–588. Springer.

[7] Boldyreva, A., Goyal, V. and Kumar, V. (2008) Identity-based Encryption with Efficient Revocation. *Proc. 2008 ACM Conf. Computer and Communications Security, CCS 2008*, Alexandria, VA, USA, October 27–31, pp. 417–426. ACM.

[8] Attrapadung, N. and Imai, H. (2009) Attribute-based Encryption Supporting Direct/indirect Revocation Modes. *Proc. 12th IMA Int. Conf. Cryptography and Coding 2009*, Cirencester, UK, December 15–17, Lecture Notes in Computer Science 5921, pp. 278–300. Springer.

[9] Attrapadung, N. and Imai, H. (2009) Conjunctive Broadcast and Attribute-based Encryption. *Proc. 3rd Int. Conf. Pairing-Based Cryptography – Pairing 2009*, Palo Alto, CA, USA, August 12–14, Lecture Notes in Computer Science 5671, pp. 248–265. Springer.

[10] Lewko, A.B., Okamoto, T., Sahai, A., Takashima, K. and Waters, B. (2010) Fully Secure Functional Encryption: Attribute-based Encryption and (hierarchical) Inner product Encryption. *Proc. 29th Annual Int. Conf. Theory and Applications of Cryptographic Techniques, Advances in Cryptology – EUROCRYPT 2010*, French Riviera, May 30–June 3, Lecture Notes in Computer Science 6110, pp. 62–91. Springer.

[11] Waters, B. (2009) Dual System Encryption: Realizing Fully Secure IBE and HIBE under Simple Assumptions. *Proc. 29th Annual Int. Cryptology Conf., Advances in Cryptology – CRYPTO 2009*, Santa Barbara, CA, USA, August 16–20, Lecture Notes in Computer Science 5677, pp. 619–636. Springer.

[12] Goyal, V., Pandey, O., Sahai, A. and Waters, B. (2006) Attribute-based Encryption for Fine-grained Access Control of Encrypted Data. *Proc. 13th ACM Conf. Computer and Communications Security, CCS 2006*, Alexandria, VA, USA, October 30–November 3, Lecture Notes in Computer Science 5126, pp. 89–98. Springer.

[13] Bethencourt, J., Sahai, A. and Waters, B. (2007) Ciphertext-policy Attribute-based Encryption. *2007 IEEE Symp. Security and Privacy (S&P 2007)*, Oakland, CA, USA, May 20–23, pp. 321–334. IEEE Computer Society.

[14] Ostrovsky, R., Sahai, A. and Waters, B. (2007) Attribute-based Encryption with Non-monotonic Access Structures. *Proc. 2007 ACM Conf. Computer and Communications Security, CCS 2007*, Alexandria, VA, USA, October 28–31, pp. 195–203. ACM.

[15] Goyal, V., Jain, A., Pandey, O. and Sahai, A. (2008) Bounded Ciphertext Policy Attribute based Encryption. *Proc. 35th Int. Colloquium, Automata, Languages and Programming, ICALP 2008, Part II – Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, Reykjavik, Iceland, July 7–11, pp. 579–591.

[16] Waters, B. (2011) Ciphertext-policy Attribute-based Encryption: An Expressive, Efficient, and Provably Secure Realization. *Proc. 14th Int. Conf. Practice and Theory in Public Key Cryptography, Public Key Cryptography – PKC 2011*, Taormina, Italy, March 6–9, Lecture Notes in Computer Science 6571, pp. 53–70. Springer.

[17] Lewko, A.B. and Waters, B. (2012) New Proof Methods for Attribute-based Encryption: Achieving Full Security through Selective Techniques. *Proc. 32nd Annual Cryptology Conf., Advances in Cryptology – CRYPTO 2012*, Santa Barbara, CA, USA, August 19–23, Lecture Notes in Computer Science 7417, pp. 180–198. Springer.

[18] Sahai, A., Seyalioglu, H. and Waters, B. (2012) Dynamic Credentials and Ciphertext Delegation for Attribute-based Encryption. *Proc. 32nd Annual Cryptology Conf., Advances in Cryptology – CRYPTO 2012*, Santa Barbara, CA, USA, August 19–23, Lecture Notes in Computer Science 7417, pp. 199–217. Springer.

[19] Lai, J., Deng, R.H., Yang, Y. and Weng, J. (2013) Adaptable Ciphertext-policy Attribute-based Encryption. *6th Int. Conf., Pairing-Based Cryptography – Pairing 2013, Revised Selected Papers*, Beijing, China, November 22–24, Lecture Notes in Computer Science, 8365, pp. 199–214. Springer.

[20] Boneh, D., Goh, E. and Nissim, K. (2005) Evaluating 2-dnf Formulas on Ciphertexts. *Proc. Theory of Cryptography, Second Theory of Cryptography Conf., TCC 2005*, Cambridge, MA, USA, February 10–12, Lecture Notes in Computer Science 3378, pp. 325–341. Springer.

[21] Naor, D., Naor, M. and Lotspiech, J. (2001) Revocation and Tracing Schemes for Stateless Receivers. *Proc. 21st Annual Int. Cryptology Conf., Advances in Cryptology – CRYPTO 2001*, Santa Barbara, CA, USA, August 19–23, Lecture Notes in Computer Science 2139, pp. 41–62. Springer.

[22] Freeman, D.M. (2010) Converting Pairing-based Cryptosystems from Composite-order Groups to Prime-order Groups. *Proc. 29th Annual Int. Conf. Theory and Applications of Cryptographic Techniques*, French Riviera, May 30–June 3, Lecture Notes in Computer Science 6110, pp. 44–61. Springer.

[23] Naccache, D. (2007) Secure and practical identity-based encryption. *IET Inf. Secur.*, **1**, 59–64.