Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Generic attribute revocation systems for attribute-based encryption in cloud storage*

Genlang CHEN†‡1,2, Zhiqian XU†3, Jia-jian ZHANG2, Guo-jun WANG4, Hai JIANG†5, Miao-qing HUANG6

[1] *Institute of Ningbo Technology, Zhejiang University, Ningbo 315100, China*
[2] *Ningbo Research Institute, Zhejiang University, Ningbo 315100, China*
[3] *Independent Scholar*
[4] *School of Computer Science and Technology, Guangzhou University, Guangzhou 510006, China*
[5] *Department of Computer Science, Arkansas State University, Jonesboro 72467, USA*
[6] *Department of Computer Science and Computer Engineering, University of Arkansas, Jonesboro 72467, USA*
†E-mail: cgl@zju.edu.cn; zhiqian.xu@gmail.com; hjiang@astate.edu

Received Aug. 28, 2018; Revision accepted Apr. 17, 2019; Crosschecked June 11, 2019

Abstract: Attribute-based encryption (ABE) has been a preferred encryption technology to solve the problems of data protection and access control, especially when the cloud storage is provided by third-party service providers. ABE can put data access under control at each data item level. However, ABE schemes have practical limitations on dynamic attribute revocation. We propose a generic attribute revocation system for ABE with user privacy protection. The attribute revocation ABE (AR-ABE) system can work with any type of ABE scheme to dynamically revoke any number of attributes.

Key words: Attribute-based encryption; Generic attribute revocation; User privacy; Cloud storage; Access control
https://doi.org/10.1631/FITEE.1800512 **CLC number:** TP309.2

1 Introduction

As a type of cloud service model, cloud storage inherits the benefits of cloud computing (Miller and Veiga, 2009; Carroll et al., 2011; Gibson et al., 2012) and offers a range of advantages over traditional owned storage systems. It allows consumers to pay for information technology (IT) services on a utility-like basis using a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services). However, while the benefits of cloud storage are compelling, cloud storage does have its potential downsides and risks. The unique security challenges arising from the trust in the cloud storage providers (CSPs) and the shared storage environments have special requirements for data- and user-centric access control.

Attribute-based encryption (ABE) was introduced by Sahai and Waters (2005). It is a type of public-key encryption, which has a one-to-many relationship between a public key and a set of decryption keys. Data is encrypted by a public key and a set of system parameters. Decryption keys or decryption processes are associated with an access policy. ABE does not need data owners or a third party, such as a CSP, to mediate the data access. Furthermore, when data is backed up on different servers in cloud, the access policy is transferred and enforced.

Although ABE is one of the ideal choices for data-centric protection in a cloud environment, dynamic attribute revocation is one of the practical

‡ Corresponding author

* Project supported by the Ningbo eHealth Project, China (No. 2016C11024)

limitations. In ABE, an attribute can be shared by multiple users. To revoke an attribute from a user, the data owner has to re-generate the public key and re-issue new private keys for other users to share the same attribute. The data will need to be decrypted and re-encrypted under a new key. However, we argue that any shared attribute being revoked from a user should not impact the use of the attribute by other users. In other words, attribute revocation should allow any number of attributes to be revoked from one user or multiple users without impact on other users, who share or use the same attributes for data decryption. Furthermore, no new public key or private key needs to be generated. Data does not need to be re-encrypted by a new key.

Attribute revocation has been studied in the literature; however, most existing revocation schemes work only with certain individual ABE schemes. Some schemes are inflexible in the number of attributes that can be revoked, or not granular enough to support attribute revocation at both user and attribute levels (two-level revocation): revoking any number of attributes from a user or revoking any attribute from a number of users. We propose an attribute revocation system that is generic to all ABE schemes and supports two-level attribute revocation. User privacy is protected in the revocation system.

2 Related work

Yu et al. (2010) proposed a ciphertext policy (CP) ABE scheme to accomplish revocation of user access rights via attribute revocation. When a user's access right is revoked, the ABE attribute authority (AA) generates a new re-encryption key for the semi-trusted online proxy server. On behalf of the AA, the proxy server generates and distributes newly updated attribute key shares (private keys) to each non-revoked user. Then the proxy server re-encrypts the data with a new encryption key. Although the scheme offloads the data and the attribute private key updates to a semi-trusted proxy server, each revocation triggers a round of user attribute key updates and data re-encryption.

Yang et al. (2013) proposed a CP-ABE revocation scheme, which generates a new attribute public key and a private key whenever an attribute is revoked. The difference between this scheme and the one proposed by Yu et al. (2010) is that only those components associated with the revoked attribute in secret keys and ciphertexts, instead of all the components in secret keys and ciphertexts, need to be updated. Users are required to obtain those updates of their private keys. Data is re-encrypted by semi-trusted CSPs using the new attribute public key. While these schemes enable instantaneous user revocation, each revocation still triggers a round of attribute key share updates and ciphertext re-encryption.

Based on the user revocation system proposed by Hur and Noh (2011), Xie et al. (2013) constructed a user and attribute revocation system, aiming at reducing the computational overhead of the data service manager in Hur and Noh (2011). The new construction proposed by Xie et al. (2013) improved the key update computation of the data service manager by half. Despite the efficient new construction, it inevitably inherits the potential management overhead of service managers in Hur and Noh (2011). The attribute key encryption keys (KEKs) are generated and maintained via a global binary tree that assigns users to the leaf nodes. For a large group of users, maintaining the binary tree becomes hard when the system needs to add or delete users. The data service manager has to know every user's attribute set to generate and distribute their attribute KEKs, which requires trust in the data service manager. This kind of trust does not exist in the untrusted cloud storage environments. In addition, every user needs to have two sets of keys: secret attribute key shares and attribute KEKs.

Wang et al. (2011) proposed an attribute revocation for a key policy (KP) ABE scheme. In the scheme, each user is associated with two access trees. Each tree has its own private key. The first tree is used for data decryption if the user does not have any revoked attributes. Otherwise, the second tree is used as long as the user's valid attributes still satisfy the policy of the KP-ABE scheme. Although this revocation scheme does not require attribute key re-issuing, it works only for revoking one attribute at a time. Naruse et al. (2014) proposed an attribute revocation scheme to CP-ABE schemes using a cloud server by proxy re-encryption to revoke attributes. This scheme does not require generations of the new secret key when granting attributes to a user. However, this revocation can apply to only CP-ABE schemes built on the linear secret sharing scheme (LSSS) access structure. Imine et al. (2017) proposed an attribute revocation mechanism for a decentralized CP-ABE scheme. Their revocation approach built on an existing decentralized CP-ABE scheme can immediately revoke an attribute without redistributing user keys. However, this revocation is still limited to be applied to a particular scheme. Xue et al. (2018) proposed a KP-ABE scheme with an attribute revocation feature, AD-KP-ABE, to protect data from being accidently deleted in the cloud storage. The construction used the attribute revocation cryptographic primitive and Merkle Hash tree to achieve fine-grained access control and verifiable data deletion. Although AD-KP-ABE can revoke an attribute without secret key update, it works only for its own proposed KP-ABE scheme.

3 Contributions

We propose an attribute revocation ABE (AR-ABE) system for ABE schemes with the following characteristics:

1. Generic

The revocation process can directly work with any ABE scheme.

2. Dynamic

The revocation is instantaneous without requiring ABE private keys be re-issued or data be re-encrypted.

3. Granular

The revocation can take place at either the attribute level or the user level. At the attribute level, an attribute can be revoked from a user or a number of users. At the user level, a user can have one attribute or multiple attributes being revoked. Any level of revocation does not impact other users using the same attributes.

4. Collusion resistant for re-encryption keys

The revocation system uses ciphertext re-encryption to control the access to ABE ciphertexts. Ciphertext re-encryption keys are re-constructed by users with their attribute witnesses. The AR-ABE system prevents users from pooling their attribute witnesses to construct any ciphertext re-encryption key. Re-encryption keys are completely independent with regard to ABE private keys.

5. Anonymous

Users do not send their identity information that can individually identify them to CSP when retrieving encrypted data. Beyond the attributes dictate in the policy attached to the ciphertext, CSP cannot identify any user. Thus, users are anonymous to CSP.

In the following sections, we first introduce cryptographic preliminaries that will be used for the security analysis of AR-ABE. Then we describe and design an AR-ABE system. Data structures and system flow diagrams are used for explanation. As AR-ABE is a secure data protection system rather than a cryptographic scheme, we analyze the security of the system. Finally, we inspect the overhead of adding generic attribute revocation to an ABE scheme.

4 Cryptographic preliminaries

As AR-ABE is a system design instead of a cryptographic scheme, security will be analyzed instead of formally approved. We provide a brief background of cryptography necessary for the remainder of this study.

4.1 Diffie-Hellman assumptions

The Diffie-Hellman assumptions are closely related to the difficulty of computing the discrete logarithm problem over a cyclic group (McCurley, 1990).

Let IG be a polynomial-time algorithm that takes security parameters 1^n as inputs and outputs the tuple $\langle G,\ p,\ g\rangle$, where G is a cyclic group, p is the order of G, and g is a generator of G. The discrete logarithm problem (DLP) in $\langle G,\ p,\ g\rangle$ is defined as follows:

Given $(g,\ g^a)$, compute a, where a is randomly chosen from Z_p. A polynomial-time adversary A has an advantage of ε in solving the DLP in $\langle G,\ p,\ g\rangle$ if $\Pr\left[A(g,\ g^a) = a\right] \geq \varepsilon$, where the probability is over the random choice of $a \in Z_p$ and the random bits used by A.

A group G satisfies the DLP if there is no such adversary A with a non-negligible advantage.

The computational Diffie-Hellman (CDH) assumption (Boneh, 1998) in $\langle G,\ p,\ g\rangle$ is defined as follows:

Given (g, g^a, g^b), compute g^{ab}, where a and b are randomly selected from Z_p. A polynomial-time adversary A has an advantage of ε in solving the CDH problem in $\langle G,\ p,\ g\rangle$ if $\Pr\left[A(g,\ g^a,\ g^b) = g^{ab}\right] \geq \varepsilon$, where the probability is over the random choice of

$a,\ b \in Z_p$ and the random bits used by A.

We say that a group G satisfies the CDH, if there is no such adversary A with a non-negligible advantage.

The decisional Diffie-Hellman (DDH) assumption (Boneh, 1998) in $\langle G,\ p,\ g\rangle$ is defined as follows:

Given $(g,\ g^a,\ g^b,\ g^c)$, determine whether $c = ab$, where $a,\ b$, and c are randomly selected from Z_p. A polynomial-time adversary A has an advantage of ε in solving the DDH problem in $\langle G,\ p,\ g\rangle$, if

$$\varepsilon \leq \Big| \Pr\left[A(g,\ g^a,\ g^b,\ g^{ab}) = 1\right] - \Pr\left[A(g,\ g^a,\ g^b,\ g^c) = 1\right] \Big|,$$

where the probability is over the random choice of $a,\ b$, and $c \in Z_p$, and the random bits used by A.

We say that a group G satisfies the DDH, if there is no such adversary A with a non-negligible advantage.

4.2 Bilinear maps and Diffie-Hellman assumptions

Many cryptographic schemes have been constructed using bilinear maps (Joux, 2000; Verheul, 2001), such as the CP-ABE scheme (Bethencourt et al., 2007). Computational complexity assumptions for bilinear maps, such as bilinear Diffie-Hellman assumptions, are thus used for arguing the security of the schemes. We will use bilinear maps to construct our security systems.

4.2.1 Definition of bilinear maps

Let $G_1,\ G_2$, and G_T be multiplicative cyclic groups of prime order p. Let g_1 and g_2 be generators of G_1 and G_2, respectively. A mapping $e: G_1 \times G_2 \rightarrow G_T$ is said to be bilinear if it has the following properties (Boneh et al., 2004):

1. Bilinearity

For all $u \in G_1,\ v \in G_2$, and a and $b \in Z_p$, $e(u^a,\ v^b) = e(u,\ v)^{ab}$.

2. Non-degeneracy

$e(g_1,\ g_2) \neq 1$.

In this study, we consider $G_1 = G_2$, or $e: G \times G \rightarrow G_T$. The existence of a bilinear map $e: G \times G \rightarrow G_T$ has two direct implications for the groups involved:

(1) Menezes, Okamoto, and Vanstone (MOV) reduction reduction (Menezes et al., 1993)

The DLP is no harder in G than in G_T. To see this, let P and $Q \in G$. We would like to find $a \in Z_p$ such that $Q = P^a$. Let $g = e(P,\ P)$ and $h = e(P,\ Q)$. Then we have $h = g^a$. Therefore, we reduce the DLP in G to DLP in G_T.

(2) DDH that is easy (Joux and Nguyen, 2003)

The DDH in G is to distinguish between distributions of $(g,\ g^a,\ g^b,\ g^{ab})$ and $(g,\ g^a,\ g^b,\ g^c)$, where $a,\ b$, and $c \in Z_p$ are randomly selected. The DDH in G_T is easy. To see this, observe that: Given $g,\ g^a,\ g^b$, and $g^c \in G^{\star}$, we have $c = ab \bmod q \Longleftrightarrow e(g,\ g^c) = e(g^a,\ g^b)$.

5 Generic attribute revocation system (AR-ABE)

Granular attribute revocation without reissuing user's private keys is not straightforward. In ABE schemes, each user's private key is uniquely formed. A special link is typically used to bind the private key shares of a user's attributes together, so that users are prevented from pooling their key shares together to construct a decryption key. This type of prevention is referred to as collusion resistance. Revoking an attribute without re-issuing the private key can break the bond among the attribute key shares, and prohibit the user from reconstructing other decryption keys. This makes it difficult to implement a dynamic and granular revocation system.

To make an attribute revocation system dynamic, granular, and generic to all types of ABE schemes, we separate revocation control from the underlying ABE schemes. This revocation control is achieved by controlling access to ABE ciphertexts via ciphertext re-encryption. Re-encryption keys are randomly generated one-time keys. If a user does not have non-revoked attributes that are required by the ABE policy, the one-time re-encryption key cannot be recovered during the decryption process, even though the user still possesses a valid ABE private key. The proposed system uses the dynamic accumulator (DA), which is built according to the scheme proposed by Au et al. (2009), to manage user's access to ABE ciphertext. Users are uniquely identified by global identifiers (gids) that are not linked to their attributes and identities.

1. At the attribute level, each attribute is associated with a set of accumulators aggregated with users (gids) who possess the attribute that has not been

revoked. When an attribute is revoked from a user, the user's gid will be removed from its accumulator. The rest of the users of the accumulator will receive their new witnesses. When a new user is granted the access to an ABE ciphertext, the user will be added to an accumulator for each attribute that the user has. Other users in those accumulators will receive new witness updates.

2. At the user level, we employ an access tree called an "attribute accumulator tree" (AATree) in Section 5.2.1.

An AATree is constructed by a data owner (DO) with attributes required by the access policy of an ABE scheme. The tree is provided to CSPs. The AATree is used for the following purposes:

(1) It contains the necessary attributes (via their accumulators) needed by the selected ABE scheme. Although AATree is inspired by the access tree of ABE schemes, it is completely independent with regard to the content and relationship with the underlying ABE scheme. An AATree is not used by any ABE scheme and does not replace any ABE scheme's access policy. It simply contains the necessary attributes that are used by the selected ABE scheme in the decryption process.

(2) It is used for controlling user's access to ABE ciphertexts. When a user requests an ABE ciphertext, a CSP randomly generates a one-time re-encryption key, re-encrypts the ABE ciphertext, and embeds the re-encryption key using the AATree in the re-encrypted ciphertext. Only users who have non-revoked attributes with their accumulator witnesses satisfying the AATree can re-construct the re-encryption key to obtain the ABE ciphertext.

(3) It is used to protect user's privacy. As the AATree and accumulators are managed and updated by DOs, CSPs do not need to know or keep a list of users with their revoked attributes. A CSP needs only to be posted with the latest AATree and uses it to embed re-encryption keys.

5.1 Trust model

As Fig. 1 shows, the trust model of the attribute revocation system for ABE in cloud storage consists of four entities:

1. AA that generates ABE private keys (or attribute key shares) for users, publishes the ABE public key $\mathrm{pk}_{\mathrm{abe}}$, and protects the ABE master secrets $\mathrm{mk}_{\mathrm{abe}}$.

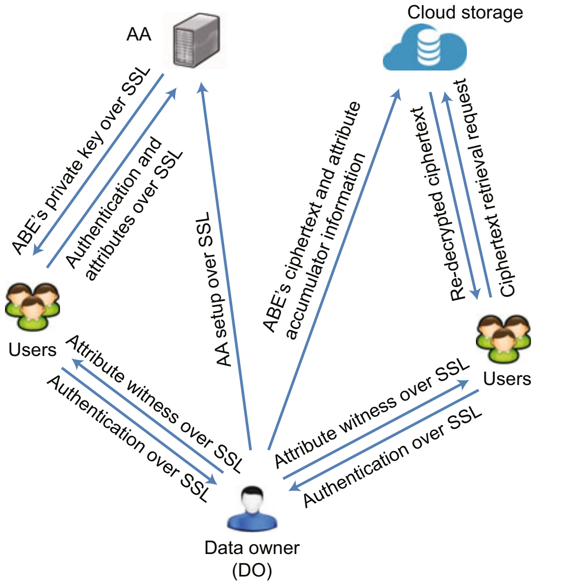

AA
Cloud storage
ABE's private key over SSL
Authentication and attributes over SSL
AA setup over SSL
ABE's ciphertext and attribute accumulator information
Re-decrypted ciphertext
Ciphertext retrieval request
Users
Users
Attribute witness over SSL
Authentication over SSL
Attribute witness over SSL
Authentication over SSL
Data owner (DO)

Fig. 1 Trust model of an attribute revocation system for ABE in cloud storage

2. DO that initializes the system, defines data access policy, and centrally manages user's access rights to the encrypted data in cloud storage.

3. CSPs provide cloud storage for a DO to store data and fulfill user's requests.

4. Users that can decrypt data only if they are eligible and have attributes complying with the access control policy of an ABE scheme.

We assume that, where necessary, communications between entities in our system are protected via suitable secure communication mechanisms, such as secure sockets layer (SSL) and transport layer security (TLS). More assumptions can be found in Chen et al. (2018).

5.2 Algorithm definitions and constructions

We first define the data structures and algorithms used in AR-ABE.

5.2.1 Data structure definition

There are two main data structures for AATree and accumulators. AATree data structure is used for managing user's access to ABE ciphertexts. Accumulator data structure is used to manage user's attributes and revocation statuses.

1. AATree

Fig. 2 shows an example of the AATree. The internal nodes are attributes that are required by the selected ABE scheme to check the compliance of its access policy. Each leaf node is one of the accumulators of the attribute. Since each user is aggregated to only one accumulator of an attribute, the internal nodes (attributes) are one-out-of-n threshold gates, where n denotes the number of accumulators for an attribute that can be different among attributes. As the number of accumulators for an attribute can be changed by adding or removing users, n is not static and is subject to change throughout its lifetime. AATree is dynamically updated by the DO. Fig. 2 illustrates how a re-encryption key is split and embedded into the tree.

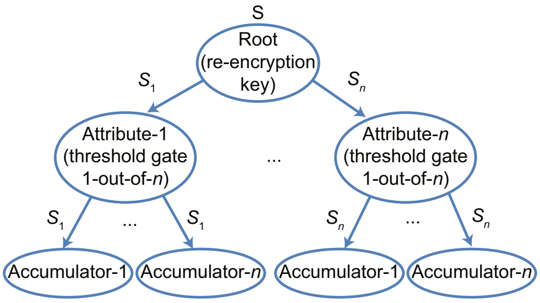


Fig. 2 Attribute accumulator tree

2. Data structures used for managing accumulators

The system and DO use data structures to manage accumulators and user attributes. Let $\mathrm{Att}=\{\mathrm{att}_i\}_{1\leq i\leq|\mathrm{Att}|}$ contain attributes in the system, where $|\mathrm{Att}|$ denotes the number of attributes in the system. Let $\varLambda_{\mathrm{policy}}=\{\mathrm{att}_i\}_{1\leq i\leq|\varLambda_{\mathrm{policy}}|}$ contain the attributes used by an ABE access policy, where $|\varLambda_{\mathrm{policy}}|$ denotes the number of attributes included. Let $\mathrm{UAtt}_{\mathrm{gid}}=\{\mathrm{att}_i\}_{1\leq i\leq|\mathrm{UAtt}_{\mathrm{gid}}|}$ contain attributes of user gid, where $|\mathrm{UAtt}_{\mathrm{gid}}|$ denotes the number of attributes of user gid. An accumulator α contains $\{v,\ \{\mathrm{gid}_x\}_{1\leq x\leq|\{\mathrm{gid}_x\}|},\ i,\ y\}$, where v denotes the aggregate value by aggregating the elements in $\{\mathrm{gid}_x\}$, $|\{\mathrm{gid}_x\}|$ denotes the number of gids in α, i is the internal identifier (also referred to as an "internal index") of α, and y is randomly selected from Z_p to generate the external identifier in the form of g^y.

Each component of α can be further denoted as follows:

$$\begin{cases}\alpha^1=v,\ \alpha^2=\{\mathrm{gid}_x\}_{1\leq x\leq|\{\mathrm{gid}_x\}|},\\ \alpha^3=i,\ \alpha^4=y.\end{cases}$$

Let $\varPhi_{\mathrm{att}}$ denote the attribute accumulator container which contains all the accumulators of attribute att, expressed as

$$\varPhi_{\mathrm{att}}=\{\varPhi^1_{\mathrm{att}}=\mathrm{att},\ \varPhi^2_{\mathrm{att}}=\varPhi=\{\alpha_i\}_{1\leq i\leq|\varPhi|}\},$$

where $|\varPhi|$ is the number of accumulators of att.

Let $\varOmega=\{\varPhi_{\mathrm{att}_i}\}_{\mathrm{att}_i\in\mathrm{Att}}$ contain all the attribute accumulator containers. Let $\mathrm{uw}_{\mathrm{gid}}=\{\mathrm{att}_i,\ w_{\mathrm{gid}},\ g^{y_j}\}_{1\leq i\leq|\mathrm{uw}_{\mathrm{gid}}|}$ contain the witnesses of the attributes that user gid has had and not been revoked.

3. Data structures used to express AATree

As accumulators are solely managed by a DO, they are kept and stored at the DO side. The information, such as the accumulator's aggregate values and attributes required by an ABE access policy, is needed by CSPs to control user's access to an ABE ciphertext. $\varDelta$ is used by a DO to provide the information for CSPs. $\varPsi$ is created by a CSP to embed re-encryption keys in the ciphertext based on $\varLambda_{\mathrm{policy}}$. $\varDelta$ is used to store accumulator aggregate values of attributes in $\varLambda_{\mathrm{policy}}$, expressed as

$$\varDelta=\{\mathrm{att}_i,\ \{v_j^{y_j},\ g^{y_j}\}_{1\leq j\leq|\varPhi^2_{\mathrm{att}_i}|}\}_{1\leq i\leq|\varLambda_{\mathrm{policy}}|},$$

where v_j is the aggregate value of accumulator α_j for attribute att_i, g^{y_j} is the external identifier of accumulator α_j, $|\varPhi^2_{\mathrm{att}_i}|=|\varPhi|$ is the total number of accumulators for att_i, and $\varPsi$ is used to carry a ciphertext re-encryption key. An example is as follows:

$$\varPsi=\{\{(e(g,g)^{s_i-\mathrm{gid}\cdot y_j\prod_{\mathrm{gid}_x\in\alpha_j}(\mathrm{gid}_x+\beta)},\ g^{y_j})\}_{1\leq j\leq\left|\varPhi^2_{\mathrm{att}_i}\right|}\}_{1\leq i\leq|\varLambda_{\mathrm{policy}}|},$$

where s_i is a randomly split share of a randomly selected s at the re-encryption time. Users re-construct re-encryption keys if their non-revoked attributes have the valid witnesses of those accumulators.

Table 1 lists the commonly used notations in this study.

5.2.2 Algorithm construction

We define the commonly used algorithms in our system. Whether the algorithms can be run privately or publicly and which information is kept secret depend on the trust model. These will be described in Section 5.3.

1. $\mathrm{Setup}_{\mathrm{abe}}(1^n)$

The algorithm takes the inputs 1^n as security parameters and initializes the ABE scheme using the security parameters 1^n. Then the public key $\mathrm{pk}_{\mathrm{abe}}$ and the master secret $\mathrm{mk}_{\mathrm{abe}}$ will be generated.

Table 1 Attribute revocation system notations

Notation	Description
c	An ABE ciphertext
AA	The ABE attribute authority
$\mathrm{Setup}_{\mathrm{abe}}$	The setup algorithm of an ABE scheme
$\mathrm{KeyGen}_{\mathrm{abe}}$	The key generation algorithm of an ABE scheme
$\mathrm{Enc}_{\mathrm{abe}}$	The encryption algorithm of an ABE scheme
$\mathrm{Dec}_{\mathrm{abe}}$	The decryption algorithm of an ABE scheme
$\mathrm{pk}_{\mathrm{abe}}$	The public key of an ABE scheme
$\mathrm{mk}_{\mathrm{abe}}$	The master secret of an ABE scheme
$\mathrm{usk}_{\mathrm{abe}}$	An ABE private key of a user
gid	A user's global unique identifier
U	The set containing user gids in the system
CSP	A cloud storage provider
DO	A data owner
$\varPhi$	An accumulator container to store the accumulators of attribute att
$\varPhi_{\mathrm{att}}$	An accumulator container to store the accumulators $\varPhi$
$\varOmega$	An accumulator container to store all the $\varPhi_{\mathrm{att}}$ in the system
k	The maximum number of elements aggregated to an accumulator
A_κ	Accumulator function to generate aggregate values
β	Trapdoor value used by the accumulator function A_k
α	A container of a particular accumulator and its information
v	An accumulator aggregate value
Ops	Accumulator operational type "Add" or "Delete"
ω_α	A witness container of α to store user's witnesses
Att	A set of attributes in the system
$\mathrm{UAtt}_{\mathrm{gid}}$	A set of attributes for user's gid
$\varLambda_{\mathrm{policy}}$	A container storing the attributes required by the ABE scheme
$\mathrm{uw}_{\mathrm{gid}}$	A container storing attribute witnesses of user gid
$\varDelta$	A data structure storing accumulator values of attributes in $\varLambda_{\mathrm{policy}}$
$\varPsi$	A data structure storing shares of a ciphertext re-encryption key
C'	A container storing the re-encrypted ABE ciphertext

2. $\mathrm{KeyGen}_{\mathrm{abe}}(\cdot)$

This is the ABE algorithm to generate user private keys. The $(\cdot)$ generically denotes the input parameters and outputs a user private key $\mathrm{usk}_{\mathrm{abe}}$.

3. UserAttributeManager (gid, Ops, $\varPhi_{\mathrm{att}}$)

This algorithm takes a user gid, the operation type Ops, and an accumulator container $\varPhi_{\mathrm{att}}$ of attribute att as the inputs. It proceeds as follows:

Extract the accumulator container $\varPhi$, expressed as

$$\varPhi = \varPhi^2_{\mathrm{att}} = \{\alpha_i\}_{1\le i\le|\varPhi^2_{\mathrm{att}}|}.$$

Replace the old $\varPhi$ in $\varPhi_{\mathrm{att}}$ with the newly returned $\varPhi$, expressed as

$$\varPhi^2_{\mathrm{att}} = \varPhi.$$

Set $w'_\alpha = \{\}$, find and replace the old α in $\varPhi^2_{\mathrm{att}}$ (using the internal index α^3), and embed gids and accumulator random number $\alpha^4(y)$ in user witnesses to prevent collusions and witness forgery. For each $\mathrm{gid}_i \in \omega_\alpha$, re-compute the witness as

$$\begin{aligned} w'_{\mathrm{gid}_i} =& \Bigg\{ {w'}^1_{\mathrm{gid}_i} = (w^1_{\mathrm{gid}_i})^{\mathrm{gid}_i\alpha^4} \\ &= g^{\frac{\mathrm{gid}_i y\left(\prod_{\mathrm{gid}_x\in\alpha^2}(\mathrm{gid}_x+\beta)\right)}{\mathrm{gid}_i+\beta}}, \\ & w^2_{\mathrm{gid}_i} = g^{(\mathrm{gid}_i+\beta)} \Bigg\}, \end{aligned}$$

and add w'_{gid_i} to ${\omega_\alpha}'$: ${\omega_\alpha}' + \{w'_{\mathrm{gid}_i},\ \mathrm{gid}_i\}$.

Finally return $\{w'_\alpha,\ \alpha,\ \varPhi_{\mathrm{att}}\}$.

4. $\mathrm{Enc}_{\mathrm{abe}}(m,\ \cdot)$

This is an ABE encryption algorithm. It takes the input of message m and the rest of parameters denoted as $(\cdot)$, and outputs an ABE ciphertext c.

5. $\mathrm{ReEnc}(c,\ \varLambda_{\mathrm{policy}},\ \varDelta,\ \mathrm{gid})$

This algorithm inputs an ABE ciphertext c, the attributes needed by the ABE access policy $\varLambda_{\mathrm{ploicy}}$, the aggregate values with external identifiers $\varDelta$, and a user's gid. It re-encrypts c as follows:

Let $\varLambda_{\mathrm{policy}} = \{\mathrm{att}_i\}_{1\le i\le|\varLambda_{\mathrm{policy}}|}$ and $\varDelta = \left\{\{\mathrm{att}_i,\ \{v_j^{y_j},\ g^{y_j}\}\}_{1\le j\le\left|\varPhi^2_{\mathrm{att}_i}\right|}\right\}_{1\le i\le|\varLambda_{\mathrm{policy}}|}$. $\varPhi_{\mathrm{att}_i}$ cannot be accessed by this algorithm. We use this denotation to express the total number of accumulators for attribute att_i. Randomly select $s \leftarrow Z_p$, re-encrypt c to $c' = ce(g,\ g)^s$, and embed s in $\varPsi$: s is split into $|\varLambda_{\mathrm{policy}}|$ shares differently and randomly. Therefore, no share is equal to any other share:

$$s = s_1 + s_2 + \cdots + s_I,$$

where $I = |\varLambda_{\mathrm{policy}}|$.

Each share is assigned to an $\mathrm{att}_i \in \varLambda_{\mathrm{policy}}$. To prevent collusion, gid is embedded into shares. As accumulators' aggregate values do not disclose the aggregated gids, gid is blindly embedded in every accumulator. However, this is not a concern. The

AATree is one-out-of-n threshold (Fig. 2). A gid can belong to only one accumulator at most. Using the accumulators' external identifiers, correct components can be located for reconstructing the key in the decryption algorithm.

The following is the steps of building $\varPsi$:

For each $\mathrm{att}_i \in \varLambda_{\mathrm{policy}}$, find $\{\mathrm{att}_i, \{v_j^{y_j},\ g^{y_j}\}_{1\leq j\leq|\varPhi^2_{\mathrm{att}_i}|}\}$ in $\varDelta$. For each accumulator of attribute att_i, compute

$$e(g^{s_i}, (v_j^{y_j})^{-\mathrm{gid}}) = e(g,g)^{s_i - \mathrm{gid}y_j\left(\prod_{\mathrm{gid}_x\in\alpha_j^2}(\mathrm{gid}_x+\beta)\right)},$$

where j is the j^{th} accumulator α_j for att_i.

The final $\varPsi$ is expressed as

$$\begin{aligned}\varPsi =&\Bigg\{\{(e(g,g)^{s_i - \mathrm{gid}y_j\left(\prod_{\mathrm{gid}_x\in\alpha_j^2}(\mathrm{gid}_x+\beta)\right)},\\ &g^{y_j})\}_{1\leq j\leq|\varPhi^2_{\mathrm{att}_i}|}\Bigg\}_{1\leq i\leq|\varLambda_{\mathrm{policy}}|}.\end{aligned}$$

Finally the algorithm outputs

$$C' = \{c' = ce(g,\ g)^s, \varPsi\}.$$

6. $\mathrm{Dec}(C',\ \mathrm{uw}_{\mathrm{gid}})$

This algorithm inputs the re-encrypted ciphertext C', and attribute witness set $\mathrm{uw}_{\mathrm{gid}}$ for the user gid. It uses the attributes as the witnesses in the $\mathrm{uw}_{\mathrm{gid}}$ to recover the re-encryption key, and decrypts C' to obtain the ABE ciphertext c.

Let $\mathrm{uw}_{\mathrm{gid}}$ be $\{\mathrm{att}_i, w_{\mathrm{gid}}, g^{y_j}\}_{1\leq i\leq|\mathrm{UAtt}_{\mathrm{gid}}|}$, where $|\mathrm{UAtt}_{\mathrm{gid}}|$ is the number of attributes of user gid, w_{gid} is the witness of attribute att_i, and g^{y_j} is the external identifier of the accumulator.

The re-encrypted key can be recovered as follows: for each att_i in $\mathrm{uw}_{\mathrm{gid}}$, use the external identifier g^{y_j} to find the re-encryption key share in $\varPsi$. If the share can be found, then proceed as follows:

(1) Compute the intermediate value using the witness as

$$\begin{aligned}e(w^1_{\mathrm{gid}}, w^2_{\mathrm{gid}}) &= e\left(g^{\frac{\mathrm{gid}y_j\left(\prod_{\mathrm{gid}_x\in\alpha_j^2}(\mathrm{gid}_x+\beta)\right)}{\mathrm{gid}+\beta}}, g^{(\mathrm{gid}+\beta)}\right)\\ &= e(g,g)^{\mathrm{gid}y_j\left(\prod_{\mathrm{gid}_x\in\alpha_j^2}(\mathrm{gid}_x+\beta)\right)}.\end{aligned}$$

(2) Recover the key share as

$$\begin{aligned}e(g,g)^{s_i} =&e(g,g)^{s_i - \mathrm{gid}y_j\left(\prod_{\mathrm{gid}_x\in\alpha_j^2}(\mathrm{gid}_x+\beta)\right)}\\ &\cdot e(g,g)^{\mathrm{gid}y_j\left(\prod_{\mathrm{gid}_x\in\alpha_j^2}(\mathrm{gid}_x+\beta)\right)}.\end{aligned}$$

Reconstruct the re-encryption key if all the shares can be recovered as

$$\prod_{1\leq i\leq|\varLambda_{\mathrm{policy}}|} e(g,g)^{s_i} = e(g,g)^s.$$

Decrypt c', which is a part of C', to obtain the ABE ciphertext as $c = c'/e(g,g)^s$. Then the algorithm returns c. Note that if the re-encryption key can be successfully constructed, c is a well-formed ABE ciphertext; otherwise, c is a random string.

7. $\mathrm{Dec}_{\mathrm{abe}}(c,\ \cdot)$

This is an ABE decryption algorithm. It takes the ciphertext c and required parameters denoted as $(\cdot)$ as inputs. $(\cdot)$ should include the user's $\mathrm{usk}_{\mathrm{abe}}$. It returns data m if $\mathrm{usk}_{\mathrm{abe}}$ can decrypt c; otherwise, the output depends on $\mathrm{Dec}_{\mathrm{abe}}$.

5.3 System description

Fig. 3 illustrates interactions and communications between a DO, an AA, a CSP, and users.

1. System setup

This is privately run by a DO. It takes in security parameters 1^n and initializes the system by setting up system parameters, an ABE scheme, and a DA. It then adds users into accumulators, issues and distributes attribute witnesses, encrypts data using an ABE, and constructs data structures.

The outputs of the process are the following:

(1) $\mathrm{pk}_{\mathrm{abe}}$ and $\mathrm{mk}_{\mathrm{abe}}$ are sent to AA. $\mathrm{pk}_{\mathrm{abe}}$ is public, and $\mathrm{mk}_{\mathrm{abe}}$ needs to be kept secret.

(2) Non-revoked attribute witnesses are sent to users. They need to be kept secret.

(3) The ABE ciphertext c, data structures $\varDelta$, and $\varLambda_{\mathrm{policy}}$ are sent to CSP. They can be public.

Detailed steps are as follows:

(1) DO calls $\mathrm{Setup}_{\mathrm{abe}}(1^n)$ to initialize an ABE scheme and a DA. The algorithm returns the ABE public key $\mathrm{pk}_{\mathrm{abe}}$, master secret $\mathrm{mk}_{\mathrm{abe}}$, and DA's system parameters $\{k,\ \beta,\ A_k,\ \varPhi\}$.

(2) DO sends AA the ABE public key $\mathrm{pk}_{\mathrm{abe}}$ and master secret $\mathrm{ms}_{\mathrm{abe}}$.

(3) DO keeps the system parameters of DA private.

(4) DO calls $\mathrm{Enc}_{\mathrm{abe}}(m,\ \cdot)$ to encrypt the message m to c.

(5) DO creates the data structure $\varLambda_{\mathrm{policy}} = \{\mathrm{att}_i\}_{1\leq i\leq|\varLambda_{\mathrm{policy}}|}$ containing all the attributes needed by the access policy of the ABE scheme.

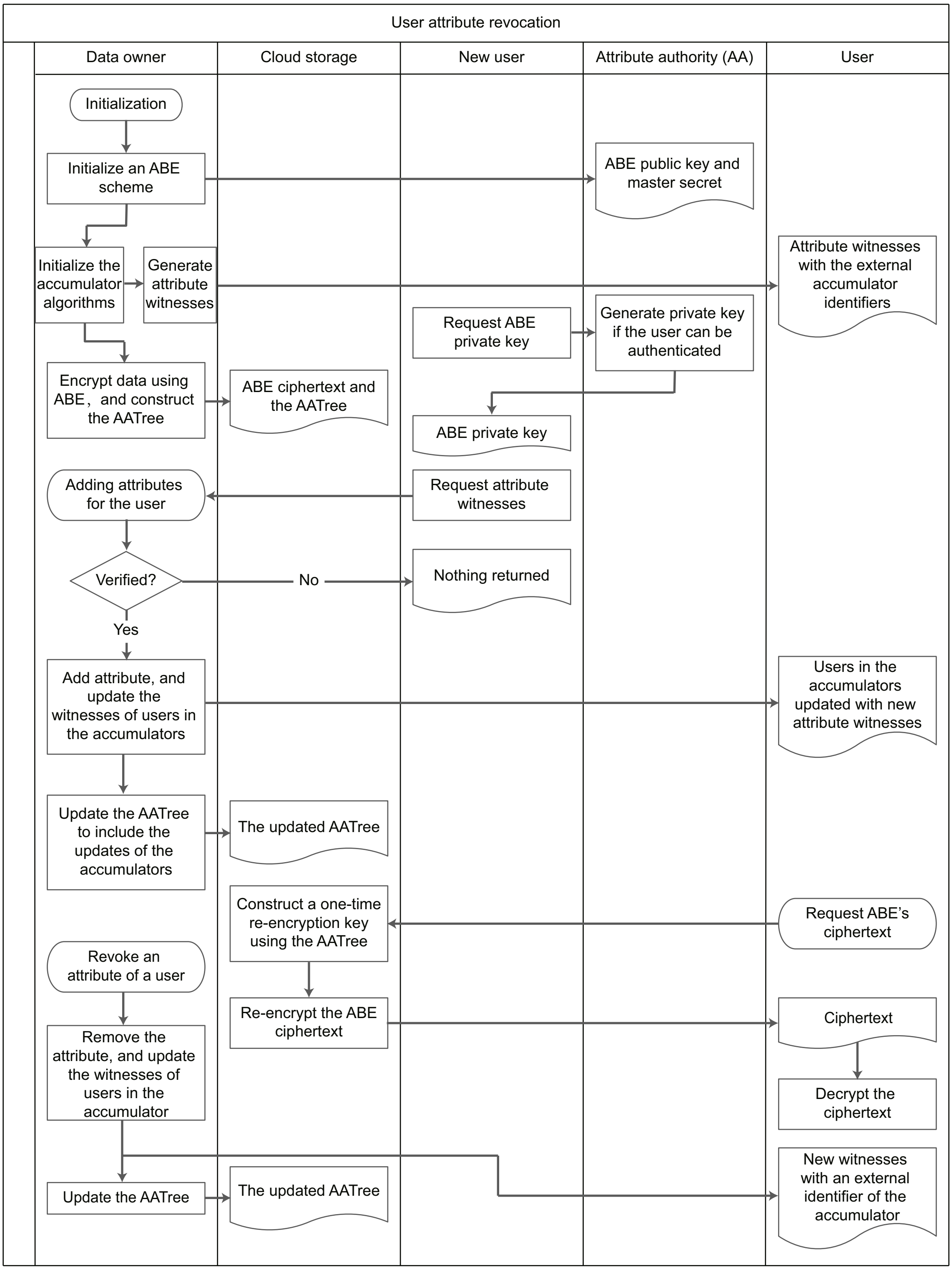

User attribute revocation
Data owner
Cloud storage
New user
Attribute authority (AA)
User
Initialization
Initialize an ABE scheme
ABE public key and master secret
Initialize the accumulator algorithms
Generate attribute witnesses
Attribute witnesses with the external accumulator identifiers
Request ABE private key
Generate private key if the user can be authenticated
Encrypt data using ABE，and construct the AATree
ABE ciphertext and the AATree
ABE private key
Adding attributes for the user
Request attribute witnesses
Verified?
No
Nothing returned
Yes
Add attribute, and update the witnesses of users in the accumulators
Users in the accumulators updated with new attribute witnesses
Update the AATree to include the updates of the accumulators
The updated AATree
Construct a one-time re-encryption key using the AATree
Request ABE's ciphertext
Revoke an attribute of a user
Re-encrypt the ABE ciphertext
Ciphertext
Remove the attribute, and update the witnesses of users in the accumulator
Decrypt the ciphertext
Update the AATree
The updated AATree
New witnesses with an external identifier of the accumulator

Fig. 3 Interaction diagram of the attribute revocation system for ABE in cloud storage

(6) DO issues attribute witnesses to users. Let U contain the existing gids. For each $\text{gid}_j \in U$, let $\text{UAtt}_{\text{gid}_j}$ contain all the attributes of gid_j. For each $\text{att}_i \in \text{UAtt}_{\text{gid}_j}$, if att_i has not been revoked, call UserAttributeManager(gid_j, "Add", Φ_{att_i}) and have $\{w_\alpha,\ \alpha,\ \Phi_{\text{att}_i}\}$ returned. If $\alpha^2 \neq \{\}$, compute the external identifier $g^{\alpha^4} = g^y$; for each $\text{gid}_i \in \alpha^2$, send gid_i the witness of att_i: $\{\text{att}_i,\ w_{\text{gid}_i}, g^y\}$.

Update Φ_{att_i} in Ω, where Ω is a container containing all the Φ_{att} in the system.

(7) DO constructs $\Delta = \{\text{att}_i,\ \{v_j^{y_j},\ g^{y_j}\}_{1\leq j\leq|\Phi^2_{\text{att}_i}|}\}_{1\leq i\leq|\Lambda_{\text{policy}}|}$ using Φ_{att_i} in Ω.

(8) DO sends c, Δ, and Λ_{policy} to a CSP.

2. A new user (gid) that requests witnesses for the attributes he/she has

This process is privately run by a DO. It first validates the user's eligibility of data access. If the user is eligible, the process adds the user to an accumulator of each non-revoked attribute. The DO then updates the attribute witnesses for all users in those changed accumulators. Δ is re-constructed at the end.

The outputs of the process are the following:

(1) Attribute witnesses in the updated accumulators are sent to the users to be kept secret.

(2) Updated Δ is sent to CSP. It can be public.

Detailed steps are as follows:

1) User gid contacts a DO and an attribute set $\text{UAtt}_{\text{gid}} = \{\text{att}_i\}_{1\leq i\leq|\text{UAtt}_{\text{gid}}|}$.

2) DO first authenticates the user and verifies the attributes in UAtt_{gid}.

3) If the user is authenticated, DO issues the witnesses as follows: For every non-revoked $\text{att}_i \in \text{UAtt}_{\text{gid}}$, call UserAttributeManager(gid, "Add", Φ_{att_i}) and have $\{w_\alpha,\ \alpha,\ \Phi_{\text{att}_i}\}$ returned. If $\alpha^2 \neq \{\}$, compute the external identifier $g^{\alpha^4} = g^y$: For each $\text{gid}_i \in \alpha^2$, send gid_i the witness update of attribute att_i: $\{\text{att}_i,\ w_{\text{gid}_i}, g^y\}$. Replace the old Φ_{att_i} with the new Φ_{att_i} in Ω.

4) The DO re-constructs $\Delta = \{\text{att}_i,\ \{v_j^{y_j},\ g^{y_j}\}_{1\leq j\leq|\Phi^2_{\text{att}_i}|}\}_{1\leq i\leq|\Lambda_{\text{policy}}|}$ using Φ_{att_i} in Ω.

5) The DO sends Δ to the CSP to replace the previous one.

3. A new user (gid) that contacts AA for an ABE's private key

This is a private process run by an AA. The output of the process is one of the following:

usk_{abe} is sent to the user and kept secret if the user is authenticated; otherwise, the output of the AA is sent to the user.

Detailed steps are as follows:

1) The user contacts an AA with the required input parameters.

2) AA authenticates the user: If the user can be authenticated, AA calls $\text{KeyGen}_{\text{abe}}(\cdot)$, generates usk_{abe}, and sends usk_{abe} to the user; otherwise, the output depends on the output of AA.

4. A user (gid) that retrieves ciphertext for decryption

This is a process run by a CSP and a user. The user contacts the CSP with his/her user gid to retrieve an ABE ciphertext c. The CSP re-encrypts c to c' using a randomly selected key. The re-encryption key is embedded into the data structure Ψ. The user is provided with c' and Ψ that are stored in C'. If the user has all the required attributes that have not been revoked, the re-encryption key can be re-constructed. Once the user obtains the ABE ciphertext c, the user calls Dec_{abe} to decrypt the data.

The output of the process is data m if the user has all the required witnesses and a valid ABE private key; otherwise, the output is the output of Dec_{abe}.

Detailed steps are as follows:

1) User gid contacts a CSP to request an ABE ciphertext c with gid.

2) CSP calls ReEnc(c, Λ_{policy}, Δ, gid) and returns $C' = \{ce(g,\ g)^s,\ \Psi\}$ to the user.

3) Let $\text{uw}_{\text{gid}} = \{\text{att}_{\text{i}}\ w_{\text{gid}},\ g^{y_j}\}_{1\leq i\leq|\text{UAtt}_{\text{gid}}|}$ denote the container who has all the attribute witnesses for gid.

4) The user calls $\text{Dec}(C', \text{uw}_{\text{gid}})$. If uw_{gid} has all the required and valid attribute witnesses, the ABE ciphertext c is returned; otherwise, a random string is returned.

5) User gid calls $\text{Dec}_{\text{abe}}(c, \cdot)$ to decrypt m, if c is correctly returned and usk_{abe} can decrypt c according to the access policy of the ABE; otherwise, the output is the output of Dec_{abe}.

5. DO that revokes an attribute of user gid

This is a private process run by a DO. The process takes the user gid and the attribute att to be revoked as the inputs. It first removes the gid from an accumulator for att, and then updates the rest of users in the accumulator with new attribute witnesses. It reconstructs Δ and sends it to CSP.

The outputs of the process are as follows: Attribute witnesses in the updated accumulator are sent to the users and kept secret. Re-constructed Δ is sent to CSP and can be public.

Detailed steps are as follows:

1) The DO extracts the attribute accumulator container Φ_{att} of att from Ω.

2) The DO calls UserAttributeManager(gid, "Delete", Φ_{att}) and has $\{w_\alpha, \alpha, \Phi_{\text{att}}\}$ returned.

3) If $\alpha^2 \neq \{\}$, the DO computes the external identifier $g^{\alpha^4} = g^y$. For each $\text{gid}_i \in \alpha^2$, send the updated witness $\{\text{att}, w_{\text{gid}_i}, g^y\}$ to user gid_i.

4) The DO replaces the old Φ_{att} with a new Φ_{att} in Ω.

5) The DO re-constructs $\Delta = \{\text{att}_i, \{v_j^{y_j}, g^{y_j}\}_{1\leq j\leq |\Phi^2_{\text{att}_i}|}\}_{1\leq i\leq |\Lambda_{\text{policy}}|}$ using Φ_{att_i} in Ω.

6) Finally, the DO sends Δ to the CSP to replace the previous one.

5.4 Security analysis

The goal of the generic attribute revocation system is to prevent users from using their revoked attributes to decrypt ABE ciphertexts after revocations. As the revocation capability is built on the top of an ABE scheme and leverages a dynamic accumulator (DA) scheme (Au et al., 2009) to achieve dynamic and anonymous user revocation, the fundamental security of the system first relies on the security of the selected ABE scheme and DA scheme. Assume that the ABE scheme is at least chosen plaintext attack (CPA) secure (Katz and Lindell, 2014), and that the DA scheme is secure against any forgeability of the witnesses. The DA provides anonymity protection for elements aggregated into the accumulators.

The attribute revocation capability aims to equip an ABE scheme with two additional security features:

1. Any number of attributes revoked from one user or multiple users does not impact other users using the same attribute or attributes. The users can still use their non-revoked attributes despite their other attributes being revoked. ABE private key is not required to be re-generated after a revocation.

2. Users can anonymously request re-encrypted ABE ciphertexts from CSPs.

With those in mind, the system should meet the following security requirements:

1. Dynamic attribute revocation

Users are prevented from using their revoked attributes to decrypt ABE ciphertexts, even when they hold valid ABE private keys. However, this protection applies to the ciphertexts retrieved after revocation.

2. Witness unforgeability

Users should not be able to forge their attribute witnesses.

3. Collusion prevention

A user cannot combine his/her attribute witnesses with other users' to successfully construct a re-encryption key.

4. Anonymity

Users remain anonymous to CSPs.

5. Ciphertext indistinguishability (CPA secure)

The re-encrypted ABE ciphertext is CPA secure against eavesdropping attacks.

5.4.1 Dynamic attribute revocation and collusion prevention

This security feature relies on ciphertext re-encryption to control the access of ABE ciphertext, in particular, the inability to recover ciphertext re-encryption keys using forged attribute witnesses. The witnesses unforgeability of a DA should prevent users from forging any attribute witness.

We use the following scenario to informally analyze that a user is prevented from using revoked attribute(s) to recover any re-encryption key. The system is assumed to be concerned with probabilistic polynomial time (PPT) adversaries.

1. Assume that user gid_A has a set of attributes, which can be denoted by $\text{UAtt}_{\text{gid}_A} = \{\text{att}_i\}_{1\leq i\leq |\text{UAtt}_{\text{gid}_A}|}$. They are not revoked at the beginning. gid_A is given the witnesses of his/her attributes of $\{\text{att}_i, w_{\text{gid}}, g^{y_j}\}_{1\leq i\leq |\text{UAtt}_{\text{gid}_A}|}$.

2. Suppose that user gid_A has attribute $\text{att}_j \in \Lambda_{\text{policy}}$ revoked. The DO removes gid_A from the accumulator for attribute att_j and re-generates the witnesses for the remaining users in that accumulator. The DO updates and sends CSP: $\Delta = \{\text{att}_i, \{v_j^{y_j}, g^{y_j}\}_{1\leq j\leq |\Phi_{\text{att}_i}|}\}_{1\leq i\leq |\Lambda_{\text{policy}}|}$, which contains the accumulator's updated aggregate values.

3. User gid_A can freely ask for any number of re-encrypted ciphertexts from CSP within polynomial time, each time the re-encryption key is randomly generated and embedded in Ψ. Since attribute att_j for user gid_A is not a member of any accumulator in Δ after its revocation, gid_A does not have the valid

witness of an accumulator for att_j.

(1) Suppose that user gid_A still holds the old witness of att_j. Based on the membership of the unforgeability (Au et al., 2009), user gid_A should not be able to forge any valid witness of any accumulator if gid_A is not aggregated to any accumulator for att_j.

(2) Suppose that user gid_A tries to re-construct the re-encryption key either using his/her old attribute witness or colluding with other users (assuming that α_i, one of the accumulators for att_j, has gid_A removed).

As described in the Dec algorithm, the following steps recover a share of the re-encryption key: The external identifier g^{y_i} provided in the witness of the attribute of $\left\{\mathrm{att}_j,\ w_{\mathrm{gid}_A} = \left(g^{\frac{\mathrm{gid}_A y_i\left(\prod_{\mathrm{gid}_x\in\alpha_i^2}(\mathrm{gid}_x+\beta)\right)}{\mathrm{gid}_A+\beta}}, g^{\mathrm{gid}_A+\beta}\right), g^{y_i}\right\}$ is used to locate the share in Ψ.

Then a share for this attribute att_j can be recovered as follows:

$$\begin{cases} e(w^1_{\mathrm{gid}_A}, w^2_{\mathrm{gid}_A}) = e\left(g^{\frac{\mathrm{gid}_A y_i\left(\prod_{\mathrm{gid}_x\in\alpha_i^2}(\mathrm{gid}_x+\beta)\right)}{\mathrm{gid}_A+\beta}},\right. \\ \qquad\qquad \left. g^{(\mathrm{gid}_A+\beta)}\right) \\ \qquad = e(g,g)^{\mathrm{gid}_A y_i\left(\prod_{\mathrm{gid}_x\in\alpha_i^2}(\mathrm{gid}_x+\beta)\right)}, \\ e(g,g)^{s_i} = e(g,g)^{s_i-\mathrm{gid}_A y_i\left(\prod_{\mathrm{gid}_x\in\alpha_i^2}(\mathrm{gid}_x+\beta)\right)} \\ \qquad \cdot e(g,g)^{\mathrm{gid}_A y_i\left(\prod_{\mathrm{gid}_x\in\alpha_i^2}(\mathrm{gid}_x+\beta)\right)}. \end{cases}$$

If user gid_A has the old witness, he/she does not have the accumulator's correct external identifier g^{y_i} as this identifier has been changed after his/her revocation. Since y_i is randomly selected from Z_p and kept secret by a DO, gid_A cannot guess the new g^{y_i} within the polynomial time based on the discrete logarithm problem (DLP) assumption, assuming that the prime number p is large enough. Even though user gid_A can collude with other users to obtain the new g^{y_i}, gid_A cannot forge a valid witness of an existing accumulator under the witness unforgeability of a DA.

Suppose that user gid_A can obtain user gid_B to share his/her valid witness of att_j to compute

$$e(g,g)^{\mathrm{gid}_B y_i \prod_{\mathrm{gid}_x\in\alpha_i^2}(\mathrm{gid}_x+\beta)}. \tag{1}$$

Note that α_i^2 here does not contain gid_A as his/her attribute has been revoked. However, with gid_B being embedded into the share, gid_A and gid_B cannot be cancelled in Eq. (1). Furthermore, based on the DLP assumption, A should not be able to recover the aggregate value of v_i given Eq. (1) within the polynomial time. Therefore, A cannot collude with other users to reconstruct the re-encryption key.

Assume that gid_A is the only one removed from α_i for att_j. Given g^{y_i}, gid_A cannot extract y_i under the DLP assumption. Therefore, gid_A cannot replace the old y_i with a new y_i in his/her $w^1_{\mathrm{gid}_A}$.

(3) As re-encryption key is a one-time randomly selected key, the key and shares are different each time the ABE ciphertext is requested. If gid_B can provide gid_A a share of att_j from his/her retrieved ciphertext, the share cannot work with the rest of the shares gid_A has to recover the ciphertext re-encryption key. The previous re-encryption keys recovered by gid_A cannot decrypt the newly retrieved ciphertext.

Based on the above analysis, the system achieves the dynamic user attribute revocation and collusion prevention.

5.4.2 Witness unforgeability

This protection is assumed to be provided by the scheme proposed by Au et al. (2009) upon which the DA is built.

5.4.3 Anonymity

The anonymity of a user to CSPs is provided from two aspects. A user is uniquely identified by a global identifier that is not linked to any of the user identity. An accumulator value does not leak any aggregated item within the anonymity of the DA on which it is based. Each accumulator is further randomized by a randomly selected number y.

5.4.4 Ciphertext indistinguishablilty (CPA secure)

Ciphertext sent over various channels can be eavesdropped by an adversary. Although the adversaries can obtain ABE public keys, all the re-encryption keys are one-time keys that are randomly and uniformly generated. Assume that the selected ABE scheme is CPA-secure. Intuitively, the re-encrypted ciphertext should remain CPA-secure.

The user attribute revocation control is implemented on the top of an ABE scheme, which is treated as black boxes and "wrapped" by a revocation layer. ABE private keys are independently generated from the attribute witnesses. Although an adversary may be given valid witnesses for their attributes, if those attributes do not comply with the ABE access policy, the adversary should not be able to decrypt an ABE ciphertext due to the security of the underlying ABE scheme. The control layer manages only user's access to the ABE ciphertext.

Based on the above analysis, we conclude that the proposed attribute revocation system, AR-ABE, has met the security requirements.

6 Attribute revocation overhead analysis

To the best of our knowledge, we are not aware of any similar system that can generically work with any existing ABE scheme. The analysis below is only pertaining to our system.

Adding the attribute revocation system to an ABE scheme introduces the following overhead:

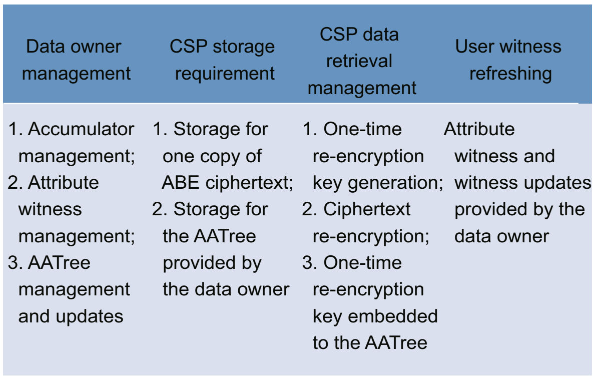
1. Accumulator management;
2. Attribute witness updates and management;
3. One-time encryption key generation and ciphertext re-encryption.

Data owners are responsible for managing the first and second overhead. The overhead potentially has scalability issues if a user's attributes or status is frequently changed. Delegating management tasks to a trusted entity and servers can dramatically improve the scalability. The 3rd overhead is at CSPs and takes place whenever a CSP fulfills a data retrieval request. This can become a bottleneck during a peak time of data retrieval. However, parallel processing in the re-encrypting ciphertext and constructing the data structure to carry the re-encryption key can help ease the bottleneck.

Fig. 4 shows the overhead incurred by different entities in the system.

7 Conclusions

In this study, we have proposed an AR-ABE system for ABE schemes. Our system is generic, and can be directly applied to any ABE scheme without modifying the underlying ABE scheme. The revocation is dynamic. It neither requires the ABE scheme re-issue user private keys, nor requires it re-encrypt the message for any revoked attribute. There is no limitation on the number of attributes to be revoked from a user or multiple users. AR-ABE protects user privacy, and thus is practical for deployment in the untrusted cloud storage environments.



Data owner management	CSP storage requirement	CSP data retrieval management	User witness refreshing
1. Accumulator management; 2. Attribute witness management; 3. AATree management and updates	1. Storage for one copy of ABE ciphertext; 2. Storage for the AATree provided by the data owner	1. One-time re-encryption key generation; 2. Ciphertext re-encryption; 3. One-time re-encryption key embedded to the AATree	Attribute witness and witness updates provided by the data owner

Fig. 4 Overhead of the attribute revocation system (AR-ABE)

Compliance with ethics guidelines

Genlang CHEN, Zhiqian XU, Jia-jian ZHANG, Guo-jun WANG, Hai JIANG, and Miao-qing HUANG declare that they have no conflict of interest.

References

Au MH, Tsang PP, Susilo W, et al., 2009. Dynamic universal accumulators for DDH groups and their application to attribute-based anonymous credential systems. In: Fischlin M (Ed.), Topics in Cryptology-CT-RSA. Springer Berlin, Germany, 5473:295-308. https://doi.org/10.1007/978-3-642-00862-7_20

Bethencourt J, Sahai A, Waters B, 2007. Ciphertext-policy attribute-based encryption. Proc IEEE Symp on Security and Privacy, p.321-334. https://doi.org/10.1109/SP.2007.11

Boneh D, 1998. The decision Diffie-Hellman problem. 3rd Algorithmic Number Theory Symp, 1423:48-63. https://doi.org/10.1007/BFb0054851

Boneh D, Boyen X, Shacham H, 2004. Short group signatures. In: Franklin M (Ed.), Advances in Cryptology-CRYPTO. Springer Berlin, Germany, 3152:227-242. https://doi.org/10.1007/978-3-540-28628-8_3

Carroll M, van der Merwe A, Kotzé P, 2011. Secure cloud computing: benefits, risks and controls. Information Security South Africa, p.1-9. https://doi.org/10.1109/ISSA.2011.6027519

Chen GL, Xu ZQ, Jiang H, et al., 2018. Generic user revocation systems for attribute-based encryption in cloud storage. *Front Inform Technol Electron Eng*, 19(11):1362-1384. https://doi.org/10.1631/FITEE.1800405

Gibson J, Rondeau R, Eveleig D, et al., 2012. Benefits and challenges of three cloud computing service models. 4th Int Conf on Computational Aspects of Social Networks, p.198-205. https://doi.org/10.1109/CASoN.2012.6412402

Hur J, Noh DK, 2011. Attribute-based access control with efficient revocation in data outsourcing systems. *IEEE Trans Parall Distrib Syst*, 22(7):1214-1221. https://doi.org/10.1109/TPDS.2010.203

Imine Y, Lounis A, Bouabdallah A, 2017. Immediate attribute revocation in decentralized attribute-based encryption access control. IEEE Trustcom/BigDataSE/ICESS, p.33-40. https://doi.org/10.1109/Trustcom/BigDataSE/ICESS.2017.217

Joux A, 2000. A one round protocol for tripartite Diffie-Hellman. In: Bosma W (Ed.), Algorithmic Number Theory. Springer Berlin, Germany, 1838:385-393. https://doi.org/10.1007/10722028_23

Joux A, Nguyen K, 2003. Separating decision Diffie-Hellman from computational Diffie-Hellman in cryptographic groups. *J Cryptol*, 16(4):239-247. https://doi.org/10.1007/s00145-003-0052-4

Katz J, Lindell Y, 2014. Introduction to Modern Cryptography (2nd Ed.). Chapman and Hall/CRC, Boca Raton, America.

McCurley KS, 1990. The discrete logarithm problem. Proc Symp in Applied Mathematics, p.49-74.

Menezes A, Okamoto T, Vanstone SA, 1993. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Trans Inform Theory*, 39(5):1636-1646. https://doi.org/10.1109/18.259647

Miller HG, Veiga J, 2009. Cloud computing: will commodity services benefit users long term? *IT Prof*, 11(6):57-59. https://doi.org/10.1109/MITP.2009.117

Naruse T, Mohri M, Shiraishi Y, 2014. Attribute-based encryption with attribute revocation and grant function using proxy re-encryption and attribute key for updating. In: Park J, Stojmenovic I, Choi M, et al. (Eds.), Future Information Technology. Springer Berlin Heidelberg, 276:119-125. https://doi.org/10.1007/978-3-642-40861-8_18

Sahai A, Waters B, 2005. Fuzzy identity-based encryption. In: Cramer R (Ed.), Advances in Cryptology-EUROCRYPT. Springer Berlin, Germany, 3494:457-473. https://doi.org/10.1007/11426639_27

Verheul ER, 2001. Self-blindable credential certificates from the weil pairing. In: Boyd C (Ed.), Advances in Cryptology-ASIACRYPT. Springer Berlin, Germany, 2248:533-551. https://doi.org/10.1007/3-540-45682-1_31

Wang PP, Feng DG, Zhang LW, 2011. Towards attribute revocation in key-policy attribute based encryption. In: Lin D, Tsudik G, Wang X (Eds.), Cryptology and Network Security. Springer Berlin, Germany, 7092:272-291. https://doi.org/10.1007/978-3-642-25513-7_19

Xie XX, Ma H, Li J, et al., 2013. New ciphertext-policy attribute-based access control with efficient revocation. In: Mustofa K, Neuhold EJ, Tjoa AM, et al. (Eds.), Information and Communication Technology. Springer Berlin, Germany, 7804:373-382. https://doi.org/10.1007/978-3-642-36818-9_41

Xue L, Yu Y, Li YN, et al., 2018. Efficient attribute-based encryption with attribute revocation for assured data deletion. *Inform Sci*, 479:640-650. https://doi.org/10.1016/j.ins.2018.02.015

Yang K, Jia XH, Ren K, 2013. Attribute-based fine-grained access control with efficient revocation in cloud storage systems. Proc 8th ACM SIGSAC Symp on Information, Computer and Communications Security, p.523-528. https://doi.org/10.1145/2484313.2484383

Yu SC, Wang C, Ren K, et al., 2010. Attribute based data sharing with attribute revocation. Proc 5th ACM Symp on Information, Computer and Communications Security, p.261-270. https://doi.org/10.1145/1755688.1755720