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

Genlang CHEN‡[1], Zhiqian XU[2], Hai JIANG[3], Kuan-ching LI[4]

[1]*Institute of Ningbo Technology, Zhejiang University, Ningbo 315000, China*

[2]*Independent Scholar*

[3]*Department of Computer Science, Arkansas State University, Jonesboro 72467, USA*

[4]*Department of Computer Science and Information Engineering, Providence University, Taiwan 43301, China*

E-mail: cgl@zju.edu.cn; zhiqian.xu@gmail.com; hjiang@astate.edu; kuancli@pu.edu.tw

Received June 27, 2018; Revision accepted Nov. 11, 2018; Crosschecked Nov. 27, 2018

Abstract: Cloud-based storage is a service model for businesses and individual users that involves paid or free storage resources. This service model enables on-demand storage capacity and management to users anywhere via the Internet. Because most cloud storage is provided by third-party service providers, the trust required for the cloud storage providers and the shared multi-tenant environment present special challenges for data protection and access control. Attribute-based encryption (ABE) not only protects data secrecy, but also has ciphertexts or decryption keys associated with fine-grained access policies that are automatically enforced during the decryption process. This enforcement puts data access under control at each data item level. However, ABE schemes have practical limitations on dynamic user revocation. In this paper, we propose two generic user revocation systems for ABE with user privacy protection, user revocation via ciphertext re-encryption (UR-CRE) and user revocation via cloud storage providers (UR-CSP), which work with any type of ABE scheme to dynamically revoke users.

Key words: Attribute-based encryption; Generic user revocation; User privacy; Cloud storage; Access control
https://doi.org/10.1631/FITEE.1800405 **CLC number:** TP309.2

1 Introduction

Cloud storage is a form of distributed storage that provides massive storage resources and services to meet on-demand data center growth for corporations and remote storage for small businesses and individuals. Cloud storage is a service model provided by cloud computing (Weiss, 2007; Hayes, 2008), which is a computing paradigm using Internet-based services to support business needs. 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).

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. The potential benefits include cost savings, mobility, accessibility, reliability, availability, agility, collaboration, and sharing.

Although the benefits of cloud storage are compelling, cloud storage does have its potential downsides and risks. The unique security challenges (Chow et al., 2009; Wang et al., 2009; Gibson et al., 2012; Ren et al., 2012) arise from two major aspects: trust in the cloud storage providers (CSPs) and the shared storage environments (multi-tenancy). The

‡ Corresponding author

* Project supported by the Natural Science Foundation of Zhejiang Province, China (No. Y15F020113) and the Ningbo eHealth Project, China (No. 2016C11024)

unique security challenges have special requirements for data protection and access control.

Due to the trustworthiness of cloud storage systems, data needs to be capable of 'self-protection'. We refer to this as 'data-centric access control'. In data-centric access control, sensitive data is encrypted before it is sent to cloud storage. Thus, data-centric access control requires the following characteristics:

1. Fine-grained protection: The data access policy is defined at a data-item level. The access policy should be enforced at each access attempt, with or without involvement of the owner.

2. Dynamic access rights management: Granting or revoking of user access rights to a particular data item should be straightforward to conduct and should, ideally, be performed almost instantaneously.

3. Efficient key management: Critical key management operations, such as key distribution, key revocation, and key refreshing, should be conducted in an efficient manner that scales well, and is appropriate for the highly dynamic and heterogeneous nature of a cloud storage environment.

For user access control and management in untrusted cloud storage, user privacy protection should be considered. We refer to this as 'user-centric access control'. We argue that it should have the following characteristics:

1. Anonymous access control: Access control mechanisms should anonymously authenticate users.

2. Dynamic user management: User access granting or revocation should be anonymous, dynamic, and instantaneous to CSPs. CSPs does not need to know any user's actual identification information (PII) or administer any user information that can be directly or indirectly linked to the user's true identity.

As data is accessed by users, a data access control mechanism in untrusted cloud storage should be concerned with both the requirements of data- and user-centric controls.

The notion of attribute-based encryption (ABE) was introduced by Sahai and Waters (2005). It is a type of public-key encryption that 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. The decryption keys or decryption process is associated with an access policy. A user can encrypt data addressed to a group of users that have a specific set of requirements without knowing each individual user in the group ahead of time. Users who have the attributes satisfying the access policy can decrypt the data. All the ciphertexts or decryption keys are associated with fine-grained access policies that are automatically enforced in the decryption process.

It is always important to enforce fine-grained protection of anonymity of the participants in an untrusted environment (Wang and Huang, 2018), especially in ABE systems where the data owners and users are decoupled (Wang et al., 2014, 2015). When using ABE to protect sensitive data, access policies can be defined by data owners. Data is encrypted by a data owner based on an access policy and is stored in cloud storage. The access policy is automatically enforced during the data decryption process. 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 the cloud, the access policy is also transferred and enforced.

Although ABE is one of the ideal choices for data-centric protection in a cloud environment, dynamic user revocation is the practical limitation. An access policy in ABE defines a group of users who share the same set of attributes. When a user is revoked, the user may still hold a valid private key to decrypt the data. To remove the user's access, the data owner has to re-generate the public key and re-issue new private keys to other users in the same group. Also, the data will need to be decrypted and re-encrypted under the new key. This is not scalable in highly distributed cloud storage.

User revocation has been studied in this study. However, most current user revocation systems work only with some particular ABE schemes, and are not flexible enough to be adapted to work with general ABE schemes. User revocation is the process of removing a user's data access privilege, and in most cases, user identities are used for user identification in the rest part of revocation. In untrusted cloud storage, user identities need to be protected and separated from CSPs. Because most existing user revocation schemes have not considered user privacy protection, a data owner has to mediate every request to protect user privacy. This means that a user's request must be routed from the CSP to the data owner before the user's privilege is determined. This is not scalable. Therefore, a practical user revocation

system for ABE schemes in an untrusted cloud storage environment should be capable of anonymously identifying a revoked user without including a data owner or a trusted party in the data retrieval process.

With these aspects in mind, we have designed and built two dynamic user revocation systems using existing ABE schemes to protect data and user privacy in cloud storage. These two user revocation systems have three features:

1. Generic: Revocation capability can be directly applied to any ABE scheme.

2. Dynamic: Revocation is instantaneous with no need to re-issue any ABE private key to users.

3. Anonymous: Users are anonymous to CSPs. A revoked user is anonymously prevented from accessing data in the data retrieval or decryption process.

2 Related work

2.1 Attribute-based encryption

In their landmark work, Sahai and Waters (2005) proposed fuzzy identity based encryption (FIBE) which allows error tolerance in identity based encryption (IBE) (Boneh and Franklin, 2003) for biometric applications. Every identity is viewed as a set of descriptive attributes ω. A user can decrypt a ciphertext encrypted by identity attributes ω' if and only if his/her identity attributes ω overlap with ω' for at least a pre-determined threshold value d ($\omega \cap \omega' \geq d$). This idea of designing fine-grained data protection was later formalized as ABE.

In ABE, an attribute authority (AA) is used to issue private keys (in the form of key shares) to users. When the system is initialized, a public key, a master secret, and system parameters are generated. Data is encrypted by the public key and system parameters. The master secret is used to derive user private keys that are used to decrypt data. There are two major classes of ABE schemes: key-policy ABE (KP-ABE) and ciphertext-policy ABE (CP-ABE). In KP-ABE schemes, a user's private key is issued based on an access policy. The ciphertext is associated with a set of attributes defined by the encryptor. A user can decrypt data if the attributes associated with the data's ciphertext comply with the access policy associated with the user's private key. The idea is reversed to CP-ABE schemes, where the AA issues a private key based on the attributes held by a user. A ciphertext is associated with an access policy defined by an encryptor. A user can decrypt a ciphertext if the user has the attributes satisfying the access policy of the ciphertext.

One of the crucial security features of ABE is collusion-resistance. Since user private keys are all derived from the same master secret, it is important that each user's private key should be sufficiently randomized to prevent users from pooling their partial private keys (attribute key shares) together to decrypt any ciphertext.

The KP-ABE concept was first introduced by Goyal et al. (2006). The initial construction of Sahai and Waters (2005) can handle only one threshold gate in an access tree. The expressibility of the access policy was greatly improved in Goyal et al. (2006). Ostrovsky et al. (2007) proposed a scheme that supports a non-monotone access tree. The primary drawback of Ostrovsky et al. (2007)'s scheme is that the size of a private key can blow up with the maximum number of attributes associated with a ciphertext. Lewko et al. (2010a) improved Ostrovsky et al. (2007)'s scheme to dramatically reduce the size of private keys.

The first CP-ABE scheme was proposed by Bethencourt et al. (2007). This scheme allows an access policy expressed by a monotone tree and uses a secret sharing scheme (Shamir, 1979) in the construction. Waters (2011) took a different approach by using a linear secret sharing scheme (LSSS) (Karchmer and Wigderson, 1993) to represent an access policy. This approach was later extended by Lewko et al. (2010b) and Okamoto and Takashima (2010). Since Bethencourt et al. (2007)'s scheme was proved secure only in the generic model, multiple approaches were proposed to construct expressive and efficient CP-ABE schemes in the standard model (Goyal et al., 2008; Ibraimi et al., 2009; Lewko et al., 2010b; Waters, 2011). Non-monotone access structures in CP-ABE were studied by Cheung and Newport (2007). The initial construction was proved to be CPA-secure under the DBDH assumption, and then was improved to be CCA-secure using the Canetti-Halevi-Katz technique (Canetti et al., 2004).

The aforementioned ABE schemes focus on the expressiveness of the access policy and security. However, the size of the ciphertext can linearly grow with the increase of the number of attributes. To

improve efficiency, schemes with a constant-size ciphertext have been studied by Emura et al. (2009), Herranz et al. (2010), Attrapadung et al. (2011), and Chen et al. (2013) for threshold access trees, and Chen et al. (2011) and Attrapadung et al. (2012) for non-monotone access trees.

Multi-authority ABE (MA-ABE) schemes have been studied by Chase (2007), Lin et al. (2008), Chase and Chow (2009), Lewko et al. (2010b), Lewko and Waters (2011), Li et al. (2011), and Han et al. (2012). In an MA-ABE system, multiple AAs work together to generate user private keys, with each AA being responsible for a disjoint set of attributes. One of the biggest challenges is to prevent users from colluding. This requires that the key shares issued by different AAs be linked in a unique way for different users.

2.2 User revocation in attribute-based encryption

There has been prior research on dealing with the practical problems of implementation of ABE schemes, particularly with respect to revocation issues.

Pirretti et al. (2006) associated attributes with expiry times. This idea was enhanced by Bethencourt et al. (2007) who suggested to associate private keys (or key shares) with expiry times. Both schemes require users periodically contact the AA for new private keys (or key shares). This process has potential scalability issues and can revoke users instantly.

Junod and Karlov (2010) constructed a CP-ABE based broadcast encryption scheme that supports direct (or instant) user revocation. In their scheme, each receiver's identity is mapped to an individual attribute. The access policy consists of a set of system attributes with a set of identity attributes. Individual user revocation is achieved by updating the set of identity attributes in the access policy. This scheme is not efficient when applied to cloud storage systems because mapping each user's identity to an attribute can make the ciphertext linearly grow. It also discloses user identities to CSPs. In addition, data owners should not be directly involved in controlling data distribution after the data has been stored at the CSP.

Jahid et al. (2011) achieved user revocation with a semi-trusted proxy to participate in the decryption process. In their proposed scheme, each user obtains an identity key in addition to their attribute key shares. The identity keys are generated by a data owner using a secret sharing scheme. The data owner also generates a proxy key for the proxy, who uses the proxy key to transfer the ciphertext in such a way that only non-revoked users with their identity keys can decrypt the data. The proxy key is regenerated whenever a user is revoked. Although the scheme achieves dynamic user revocation without attribute key regeneration, it can revoke only a predefined number of users. In addition, adding a new user to the system can trigger re-keying of all the existing identity keys, which presents a potential scalability issue and some key management issues.

Hur and Noh (2011) used attribute key encryption keys (KEKs) to address user revocation for BSW's CP-ABE scheme (Bethencourt et al., 2007). Their scheme requires a data service manager (such as a CSP) to generate attribute KEKs and distribute the keys to users. The attributes in the access policy of a ciphertext are re-encrypted by their KEKs before the ciphertext is sent to a user. When a user is revoked, the impacted attribute KEKs are updated and redistributed. This approach brings potential management overhead. The attribute 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 much harder when the system needs to add or delete users. The data service manager also 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 untrusted cloud storage environments. In addition, every user needs to have two sets of keys: secret attribute key shares and attribute KEKs.

Another user revocation approach is to associate ciphertext with expiration times. In this approach, ciphertexts are encrypted with additional time stamp attributes and periodic key updates. Sahai et al. (2012) proposed a revocable storage that prevents revoked users from accessing the data once their access rights are removed. They introduced the notion of ciphertext delegation, where a ciphertext encrypted under a certain policy can be re-encrypted to a more restrictive policy using only public information. Using ciphertext delegation, the CSP can re-encrypt the ciphertexts from time t to $t+1$.

However, this mechanism requires user private keys to be periodically updated for all non-revoked users, which can potentially create a key distribution overhead in a large group of users. Users also have to keep different versions of the keys to decrypt the data.

Because most of the existing user revocation schemes work with individual schemes, we have proposed a flexible revocation model (dynamic user revocation and key refreshment, DURKR), which can work with CP-ABE schemes for dynamic user revocation and key refreshing (Xu and Martin, 2012). In the model, the master secret is split into two parts: one part is used by CP-ABE schemes to issue the attribute private keys; the other part is used to issue a proxy delegation key that is used to re-encrypt ciphertexts of CP-ABE, and thus only non-revoked users can eventually decrypt data. Although the model is intended to be flexible to work with any type of CP-ABE scheme, it requires some modification of the underlying scheme to incorporate an additional key share. In our other work (Xu and Martin, 2013), a generic framework was proposed to be adopted to any type of ABE scheme. In addition, the framework avoids the potential key escrow in ABE schemes with centralized AA settings. However, the framework does not effectively prevent revoked and non-revoked users from colluding.

The application of ABE to cloud environments has been studied. Yu et al. (2010) proposed a scheme to protect data files in a semi-trusted cloud environment. Wang et al. (2011) built a hierarchical CP-ABE (HABE) model using the notion of a hierarchical IBE (HIBE) scheme. Parno et al. (2012) constructed a verifiable computation scheme with public delegation and verifiability based on an ABE scheme. Yang et al. (2013) proposed an access control framework using a CP-ABE scheme for cloud storage systems. The framework also enables dynamic attribute revocation. Yu et al. (2008) proposed a CP-ABE scheme with a hidden access policy for untrusted content distribution networks. Their scheme also uses periodic key expiration to manage user revocation.

3 Our contribution

We have designed and constructed two data protection systems using ABE schemes in untrusted cloud storage. Because ABE protects data privacy, the newly developed systems focus on building generic user revocation mechanisms that could work with any ABE scheme and also protect user privacy during the revocation process.

To achieve generic and anonymous user revocations, we aim at controlling the user access to ABE ciphertexts. If a revoked user cannot obtain an ABE ciphertext, then the user will not be able to decrypt the data even though the user might hold a valid ABE private (decryption) key. If such control of ciphertext access is independent of ABE, then the revocation mechanism can be generically applied to any ABE scheme. With these two considerations in mind, our solution is to build an extra layer on top of an ABE scheme to control access to ABE ciphertexts. This layer, using a dynamic and anonymous accumulator technique, treats ABE as a black box. It removes the need for a data owner to mediate every data retrieval and makes users anonymous to CSPs. Using this layer, the two systems are built as follows:

1. User revocation via ciphertext re-encryption (UR-CRE): This system re-encrypts ABE ciphertexts with accumulator (group) keys, and therefore only non-revoked users can decrypt the re-encrypted ABE ciphertexts to obtain ABE ciphertexts. Users are anonymous to CSPs during the data retrieval process. The benefit of this re-encryption system is three-fold:

(1) It enables desirable user revocation control (generic, dynamic, and anonymous) for any type of ABE scheme.

(2) It protects the integrity and secrecy of the access policy embedded in ABE ciphertexts. Because access policies of CP-ABE can be parts of the ciphertexts in plaintext format, they are vulnerable to alteration and information leakage in regard to who can decrypt the data. The same issue exists in KP-ABE, where attributes are also in plaintext format.

(3) Data is self-protected. Revoked users are automatically prevented from decrypting the re-encrypted ciphertexts during the decryption process.

The drawback of this system is the overhead associated with the ciphertext re-encryption and key management of accumulators on the data owner side.

2. User revocation via cloud storage providers (UR-CSP): This system separates user revocation control from data encryption, and uses untrusted CSPs to eliminate the overhead associated

with ciphertext re-encryption. We use the anonymous accumulator mechanism to achieve the desired user verification (generic, dynamic, and anonymous), and therefore user privacy is protected during the verification process. Any revoked user is anonymously verified by CSPs and prevented from accessing ABE ciphertexts after the user's revocation.

Because ABE ciphertexts are not re-encrypted when stored at CSPs, the possible downside of this approach is that access policies included in ciphertexts are exposed to CSPs or any adversary in the unencrypted channels during transmission. The integrity and privacy of access policies are at risk.

Because accumulators are the cornerstones of building our revocation systems, we will introduce the dynamic accumulator concept first, followed by the algorithm constructions used in this study.

4 Accumulator

Accumulators were first introduced by Benaloh and de Mare (1993) as a method to condense a set of values (elements) into one value (referred to as an accumulator's aggregate value or simply an aggregate value in this study), and thus a short witness is used to demonstrate that a value has been condensed in an aggregate value. The witness has the following characteristics: (1) It is unique if the value being aggregated is unique; (2) It is infeasible to forge a witness for a value that has not been accumulated; (3) No information about the accumulated value can be leaked through the value of the witness or the aggregate value. An accumulator is called 'universal' if it supports efficient zero-knowledge membership and non-membership proofs. An accumulator is called 'dynamic universal' if the accumulator's aggregate value, membership, and non-membership witnesses can be efficiently updated.

Camenisch and Lysyanskaya (2002) introduced the notion of dynamic universal accumulator (DUA). The cost of adding or deleting an element is independent of the number of elements accumulated. Using a zero-knowledge protocol, the scheme can prove whether a committed value is in an accumulator or not. The scheme was constructed under the strong RSA assumption. A DUA based on a bilinear map was proposed by Nguyen (2005). Au et al. (2009) extended Nguyen (2005)'s work and built a DUA under the DDH assumption. The DUA has been used for anonymous credential attestation (Brickell et al., 2004).

4.1 Accumulators based on bilinear maps

The accumulator scheme used in our systems is based on the scheme of Au et al. (2009). The following algorithm descriptions directly come from Au et al. (2009):

1. Setup(1^n): This algorithm takes the security parameter 1^n as the input. It initializes and outputs system parameters as follows:

(1) Let e: $G \times G \to G_{\mathrm{T}}$ be a bilinear map of prime order p.

(2) Let A_k: $2^{Z_p^\star} \to G$ be an accumulator's function that is parametrized by a randomly selected $\beta \in Z_p^\star$: $A_k(X) = g^{\prod_{x \in X}(x+\beta)}$, where $x \in Z_p$, $A_k(X) \in G$, and β is the trapdoor or auxiliary information.

(3) Finally output the system parameters: $\{G, G_{\mathrm{T}}, e, A_k, \beta\}$.

2. MembershipVerification(w, x, v): The algorithm takes a witness w, element x, and accumulator aggregate value v as the inputs. It proves the membership of x as follows:

(1) If $e(w, g^x \cdot g^\beta) = e(v, g)$, output 'true';

(2) Otherwise, output 'false'.

3. MembershipWitness(x): This algorithm takes an element x as an input and computes the witness of x as follows:

(1) For a set of elements $Y = \{x_1, x_2, \ldots, x_n\} \in Z_p$, a membership witness for $x \in Y$ is calculated as $w = \left[g^{\prod_{i=1}^{n}(x_i+\beta)}\right]^{\frac{1}{x+\beta}}$.

(2) Output the witness: w.

4. UpdateAccumulator(x', v, Ops): The algorithm takes a new element x', an accumulator aggregate value v, and the operation type Ops ('Add' or 'Delete') as inputs. It updates v to v' as follows:

(1) If Ops = 'Add', then $v' = v^{x'+\beta}$;

(2) If Ops = 'Delete', then $v' = v^{\frac{1}{x'+\beta}}$;

(3) Output v'.

5. UpdateMembershipWitness(w, x, v, $\widehat{v}$, Ops, x'): This algorithm takes the original membership witness w of x with respect to the accumulator aggregate value v, the new aggregate value $\widehat{v}$ having x', the operation type Ops, and the element x' as inputs. It computes the new witness of x in regard to $\widehat{v}$ as follows:

(1) If Ops = 'Add', then x' is the element newly

added to $\widehat{v}$, and the new membership witness w' of x can be computed as $w' = vw^{x'-x}$;

(2) If Ops = 'Delete', then x' has been removed from v, and the new membership witness w' of x can be calculated as $w^{\frac{1}{x'-x}}\widehat{v}^{\frac{1}{x-x'}}$;

(3) Output w'.

4.2 Dynamic accumulator

We build a dynamic accumulator (DA) using the scheme proposed by Au et al. (2009) to manage user access to data. We made the following adjustments to fit our systems:

1. We remove the UpdateMemebershipWitness algorithm. Users and CSPs are not trusted with the system parameters in our trust models, and they are not allowed to update their witnesses. Data owners are responsible for centrally managing accumulators and user witness updates.

2. We want to improve the efficiency and scalability of the scheme proposed by Au et al. (2009). Like most of the existing accumulator schemes, the scheme proposed by Au et al. (2009) has two potential limitations for large-scale systems: an accumulator has a pre-defined number of elements to be aggregated, and adding or deleting an element triggers witness updates of all elements aggregated in the same accumulator. To make the scheme more practical and minimize the impact of witness updates, we use multiple accumulators to manage non-revoked users. This adjustment can be scalable in the following ways:

(1) The witness update impacts only a small numbers of users. Non-revoked users will be divided into groups. Each user belongs only to one group and is aggregated to an accumulator. When a user is revoked, the user is removed from the user's accumulator. The witness updates, and thus impacts only a smaller number of users.

(2) There is no limitation on the number of users in the system. The number of accumulators can be dynamically increased or decreased depending on the number of users in the systems.

(3) Each accumulator is associated with one internal identifier and one external identifier. The differences between the internal and external identifiers are as follows:

a) The internal identifiers (or indexes) are kept by data owners. They are static and sequential. An accumulator is deleted when all associated users have been removed. However, its index is not recycled.

b) The external identifiers are given to users and CSPs to identify or look up accumulators. They are randomized every time the associated accumulators are updated; therefore, the static index numbers are not directly exposed to users and CSPs for extra protection. Furthermore, the numbers used to randomize external identifiers have two main purposes: to randomize ciphertext re-encryption keys when ABE ciphertexts need to be re-encrypted in UR-CRE and to randomize accumulators' aggregate values each time the aggregate values are updated in UR-CSP.

To achieve better efficiency and scalability, the actual implementations can delegate the management of accumulators, especially witness updates and distribution, to a data-owner trusted third party.

Our modification does not change the mathematical and complexity theory on which DUA (Au et al., 2009) is based. Its security strength remains without relying on the number of elements that can be accumulated into an accumulator or the number of accumulators in a system.

4.2.1 Data structure definition

Table 1 lists the notations commonly used in this section.

The major data structures used by DA are defined as follows:

1. An accumulator α contains $\{v, \{\mathrm{gid}_x\}_{1\le x\le|\{\mathrm{gid}_x\}|}, i, y\}$, where v denotes the aggregate value by aggregating the elements in $\{\mathrm{gid}_x\}$ and $|\{\mathrm{gid}_x\}|$ denotes the number of gids in α, i is the internal identifier (also referred to as 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: (1) $\alpha^1 = v$; (2) $\alpha^2 = \{\mathrm{gid}_x\}_{1\le x\le|\{\mathrm{gid}_x\}|}$; (3) $\alpha^3 = i$; (4) $\alpha^4 = y$.

2. Φ stores a set of accumulators: $\Phi = \{\alpha_i\}_{1\le i\le|\Phi|}$, where $|\Phi|$ denotes the number of accumulators included.

3. w_{gid} is the witness of user gid for an accumulator. Each witness consists of two parts, w^1_{gid} and w^2_{gid}. The details can be found in Section 4.

4. ω_a is a witness container to store witnesses of the users in α: $\{\{w_{\mathrm{gid}_i}, \mathrm{gid}_i\}_{1\le i\le|\alpha^2|}\}$.

Table 1 Dynamic accumulator (DA) notation

Notation	Description
gid	A user's global unique identifier
Φ	Accumulator container containing a set of accumulators
k	Maximum number of elements aggregated into an accumulator
A_k	Accumulator function
β	Trapdoor value for the accumulator function A_k
α	Container of an accumulator and its information
v	An accumulator aggregate value
Ops	Accumulator operational type—add or delete
ω_α	Witness container storing user witnesses of accumulator α
w_{gid}	User gid's witness of an accumulator
Ind	An integer counter of the next internal index value for an accumulator

4.2.2 Algorithm construction

Our DA consists of the following algorithms. The details of their usage and information sharing among parties are specified here.

1. AccSetup(G, g): Take a group G and a generator g of G as the inputs, as shown in Algorithm 1. It defines the system parameters used by DA: the maximum number of elements k in each accumulator, the trapdoor value β, the accumulator function A_k, and the index counter Ind of internal accumulator identifiers. It initializes Φ and returns $\{k, \beta, A_k, \Phi\}$.

Algorithm 1 AccSetup

Procedure AccSetup(G, g)
1: Begin:
2: Select k to be the maximum number of elements aggregated in an accumulator.
3: Randomly select $\beta \leftarrow Z_p^\star$. /*The trapdoor of the accumulator function.*/
4: Define A_k: $2^{Z_p^\star} \rightarrow G$ to be an accumulator function. /*$A_k(\mathrm{X}) = g^{\prod_{\text{gid}_i \in X}(\text{gid}_i+\beta)}$, where g is the generator of G.*/
5: Let $v \leftarrow g$; $\alpha \leftarrow \{v, \{\}, 1, \text{null}\}$; $\Phi \leftarrow \{\alpha\}$. /*Initialize an accumulator and an accumulator container.*/
6: Set Ind = 2. /*Set the internal index counter to be the next value.*/
7: /*Note: Ind is internally used. For simplicity, it is not returned.*/
8: Return $\{k, \beta, A_k, \Phi\}$.
9: End

2. AccAdd(gid, Φ): Take a user's global identifier gid and the accumulator container Φ as the inputs, as shown in Algorithm 2. The algorithm finds the first accumulator in Φ that has fewer than k gids. A new accumulator is created if none is found. The algorithm adds the gid to the accumulator and outputs the updated accumulator and Φ.

3. AccDelete(gid, Φ): Take a user's global identifier gid and the accumulator container Φ as the inputs, as shown in Algorithm 3. It looks for gid in accumulators stored in Φ:

Algorithm 2 AccAdd

Procedure AccAdd(gid, Φ)
1: Begin:
2: Let $N = |\Phi|$. /* Obtain the number of accumulators in Φ.*/
3: Set $j = 0$.
4: For $i = 1$ to N do /* Find the first α that has fewer than k gids.*/
5: If $\left|\alpha_i^2\right| < k$ then
6: $j = i$; abort For loop. /* The accumulator is found. */
7: End if
8: End for
9: If $j == 0$ then /* No accumulator is found. Create a new one.*/
10: $j = N + 1$.
11: $\alpha_j \leftarrow$ {g, {}, Ind, null}. /*Initialize an accumulator.*/
12: Ind = Ind + 1. /*Increase the counter by 1, and thus it always points to the next value.*/
13: End if
14: /*Add gid to α_j.*/
15: /*α_j^3 is static and does not change once it is set. α_j^4 is dynamic and will be changed outside this algorithm.*/
16: $\alpha_j \leftarrow \{v_j^{\text{gid}+\beta}, \alpha_j^2 + \{\text{gid}\}, \alpha_j^3, \alpha_j^4\}$.
17: Update Φ with α_j. /*Using α_j^3, the internal index, to find and replace the existing one or add the new one.*/
18: Return $\{\alpha_j, \Phi\}$.
19: End

Algorithm 3 AccDelete

Procedure AccDelete(gid, Φ)
1: Begin:
2: Let $N = |\Phi|$. /*Obtain the number of accumulators in Φ.*/
3: Set $j = 0$.
4: For $i = 1$ to N do
5: If gid $\in \alpha_i^2$ then /*Find the accumulator having the gid.*/
6: If $|\alpha_i^2| > 1$ then
7: $v_i = v_i^{\frac{1}{\text{gid}+\beta}}$. /*Remove gid from the aggregate value α_i^1.*/
8: $\alpha_i \leftarrow \{v_i, \alpha_i^2 - \{\text{gid}\}, i, y_i\}$. /*Update the values in α_i.*/
9: Update α_i in Φ.
10: Else
11: $\alpha_i \leftarrow \{\text{g}, \{\}, N, \text{null}\}$. /*Empty the accumulator since gid is the last one.*/
12: $\Phi \leftarrow \Phi - \{\alpha_i\}$. /*Remove α_i from Φ.*/
13: End if
14: $j = i$; abort.
15: End if
16: End for
17: If $j > 0$ then
18: Return $\{\alpha_j, \Phi\}$. /*Return the accumulator. α_j can be an empty accumulator removed from Φ.*/
19: Else
20: Return $\perp$.
21: End if
22: End

(1) If an accumulator can be found, it removes the gid from the accumulator. If the gid is the only element in the accumulator, the accumulator is emptied out. Finally, it returns the accumulator and Φ.

(2) If no accumulator can be found, return $\perp$.

4. AccWitUpdate(α): Take an accumulator α as the input, as shown in Algorithm 4. It updates all the witnesses of gids in α and outputs witness set ω_α.

Algorithm 4 AccWitUpdate

Procedure AccWitUpdate(α)
1: Begin:
2: Let $\omega_\alpha = \{\}$. /*Start with an empty set.*/
3: Let $N = |\alpha^2|$. /*Set the number of gids to N.*/
4: For $i = 1$ to N /*Compute witnesses.*/
5: $w_{\text{gid}_i}^1 = v^{\frac{1}{\text{gid}_i+\beta}}$. /*$v$ is the accumulator value: α^1.*/
6: $w_{\text{gid}_i}^2 = g^{\text{gid}_i+\beta}$.
7: $w_{\text{gid}_i} = \{w_{\text{gid}_i}^1, w_{\text{gid}_i}^2\}$.
8: $\omega_\alpha \leftarrow \omega_\alpha + \{w_{\text{gid}_i}, \text{gid}_i\}$. /*Add the user's witness to ω_α.*/
9: End for
10: Return ω_α.
11: End

5 Generic user revocation systems

We have designed and constructed two generic user revocation systems based on ABE schemes in untrusted cloud storage environments. In our systems, each user is given a unique global identifier gid. Non-revoked users are randomly divided into groups, and therefore each user belongs to only one group. Each group of user gids are aggregated into an accumulator. We may interchange the terms of group and accumulator because they are one-to-one relationships with respect to a user. Each accumulator is identified by one internal index (or identifier) and one external identifier. The internal index is fixed when the accumulator is created. The external identifier is changed and randomized each time a user is added to or removed from the accumulator. Only external identifiers are given to users and CSPs to request or locate the re-encrypted ciphertext in UR-CRE, or to generate membership proofs in UR-CSP.

5.1 Algorithm definition and construction

We have defined the commonly used algorithms in our two systems. Whether the algorithms are run privately or publicly and which information should be kept secret depend on the trust models. These will be explained in sections of system design and construction of the corresponding user revocation system.

Because our systems can work with any ABE scheme, the '$\cdot$' will be used to generically denote the parameters required by ABE algorithms: $\text{Setup}_{\text{ABE}}$, $\text{KeyGen}_{\text{ABE}}$, Enc_{ABE}, and Dec_{ABE}.

Table 2 lists the commonly used notations in addition to those given in Table 1.

1. Setup(1^n): The algorithm takes the security parameter 1^n as an input and proceeds as follows:

(1) Let U contain the gids who are granted for data access: $U = \{\text{gid}_1, \text{gid}_2, \ldots, \text{gid}_n\}$, where $\text{gid}_i \in Z_p^\star$ (p is an n-bit prime).

(2) Call $\text{Setup}_{\text{ABE}}$ (1^n) with an input 1^n to initialize the ABE scheme. It outputs an ABE public key pk_{ABE} and master secret mk_{ABE}.

Table 2 User revocation system notation

Notation	Description
c	An ABE ciphertext
AA	The ABE attribute authority
$\text{Setup}_{\text{ABE}}$	The setup algorithm of an ABE scheme
$\text{KeyGen}_{\text{ABE}}$	The key generation algorithm of an ABE scheme
Enc_{ABE}	The encryption algorithm of an ABE scheme
Dec_{ABE}	The decryption algorithm of an ABE scheme
pk_{ABE}	The public key of an ABE scheme
mk_{ABE}	The master secret of an ABE scheme
usk_{ABE}	A user ABE private key
U	The data set containing non-revoked gids in the system
CSP	Cloud storage provider
DO	Data owner
wsk_{gid}	The witness private key of user gid
Λ	The container storing accumulator aggregate values (v)
c'	A re-encrypted ciphertext of c for an accumulator
C'	The container storing multiple c's

(3) Initialize the DA using the security parameter 1^n:

a) Let e: $G \times G \to G_1$ be a bilinear map for an n-bit prime p.

b) Let g be a generator of G.

c) Call AccSetup(G, g) (Algorithm 1) with G and g being the inputs. It outputs $\{k, \beta, A_k, \Phi\}$.

2. $\text{KeyGen}_{\text{ABE}}(\cdot)$: This is the ABE key generation algorithm for generating user-private keys. The '$\cdot$' denotes the input parameters. It outputs a user private key usk_{ABE} if the user is authenticated; otherwise, the output depends on the output of AA.

3. UserManager(gid, Ops, Φ): The algorithm takes a user's gid, an operation type Ops, and the accumulator container Φ as inputs. It proceeds as follows:

(1) If Ops == 'Add', call AccAdd(gid, Φ) (Algorithm 2) to add the gid to an accumulator:

$$\{\alpha, \Phi\} = \text{AccAdd}(\text{gid}, \Phi).$$

(2) If Ops == 'Delete', call AccDelete(gid, Φ) (Algorithm 3) to remove the gid from an accumulator:

$$\{\alpha, \Phi\} = \text{AccDelete}(\text{gid}, \Phi).$$

(3) If $\alpha^2 \neq \{\}$, re-randomize the external identifier and issue a new witness for each gid $\in \alpha^2$:

a) Uniformly select a random number: $y \leftarrow Z_p$. Set $\alpha^4 = y$, and update α in Φ.

We randomize an accumulator's external identifier each time when the accumulator is updated. y will be used to generate the external identifier (in the form of g^y) for α.

b) Call AccWitUpdate(α) (Algorithm 4) to recompute all witnesses for users in α^2:

$$\omega_\alpha = \text{AccWitUpdate}(\alpha).$$

(4) Return $\{\omega_\alpha, \alpha\}$ if $\alpha^2 \neq \{\}$; otherwise, return $\perp$.

4. WitnessKeyGen(ω_α, α): The algorithm takes witnesses container ω_α and accumulator α as the inputs. It generates witness keys as follows:

(1) Extract y from α^4 and compute g^y as the external identifier of α.

(2) Let witKeys = $\{\}$ be the container to store witness keys of users in α^2.

(3) For each $\text{gid}_i \in \alpha^2$:

a) Extract w_{gid_i} from ω_α.

b) Randomly select $r_i \leftarrow Z_p$ and compute the witness key as

$$\text{wsk}_{\text{gid}_i} = \left\{ \text{wsk}^1_{\text{gid}_i} = (w^1_{\text{gid}_i})^{r_i y} = g^{\frac{r_i y(\prod_{\text{gid}_x \in \alpha^2}(\text{gid}_x+\beta))}{\text{gid}_i+\beta}}, \text{wsk}^2_{\text{gid}_i} = (w_{\text{gid}_i})^{1/r_i} = g^{\frac{(\text{gid}_i+\beta)}{r_i}}, \text{wsk}^3_{\text{gid}_i} = g^y \right\}.$$

c) Add the witness key to witKeys:

$$\text{witKeys} \leftarrow \text{witKeys} + (\text{gid}_i, \text{wsk}_{\text{gid}_i}).$$

(4) Return witKeys.

5. $\text{Enc}_{\text{ABE}}(m, \cdot)$: This is the ABE encryption algorithm. It takes data m and the rest of parameters '$\cdot$' as the inputs. It encrypts data m and outputs the ciphertext c: $\{c_m, \cdot\}$.

6. ReEnc(c, Φ): The algorithm takes an ABE ciphertext c and the accumulator container Φ as the inputs. It re-encrypts c for each accumulator $\alpha_i \in \Phi$ as follows:

(1) For $\alpha_i \in \Phi$, re-encrypt c:

$$c'_i = \{c \cdot e(g^{\alpha_i^4},\ \alpha_i^1), g^{\alpha_i^4}\} = \{c \cdot e(g^{y_i},\ v_i), g^{y_i}\}$$
$$= \{c \cdot e(g,g)^{y_i \prod_{\mathrm{gid}_x \in \alpha_i^2}(\mathrm{gid}_x+\beta)}, g^{y_i}\}.$$

Since '$\cdot$' of c may contain the access policy or attributes in clear text format, there is a possibility of leaking user information, such as user attributes, to untrusted CSPs. Re-encrypting the entire c provides the protection of integrity and privacy for access policies and attributes from malicious modifications.

Although this algorithm is used only by the ciphertext re-encryption system, a possible issue to consider is that ABE ciphertext c may have different message space compared with the group G_1 used in the DA. One way to address this is to define a reversible conversion method that maps a ciphertext of ABE to a group element of G_1 in the process of re-encryption and then converts results back in the process of decryption. We will not elaborate it here. It can be addressed in the actual implementation.

(2) Finally, output $C' = \{c'_i\}_{1 \le i \le |\Phi|}$.

7. $\mathrm{Dec}(c', \mathrm{wsk}_{\mathrm{gid}})$: The algorithm takes a re-encrypted ciphertext c' and a user's witness key $\mathrm{wsk}_{\mathrm{gid}}$ as the inputs. It decrypts c' and outputs the ABE ciphertext c as follows:

(1) Decrypt ABE ciphertext c if $\mathrm{wsk}_{\mathrm{gid}}$ is a valid key:

$$c'/e(\mathrm{wsk}_{\mathrm{gid}}^1, \mathrm{wsk}_{\mathrm{gid}}^2)$$
$$= c \cdot e(g,g)^{y \prod_{\mathrm{gid}_x \in \alpha^2}(\mathrm{gid}_x+\beta)} / e(g^{\frac{r_i y (\prod_{\mathrm{gid}_x \in \alpha^2}(\mathrm{gid}_x+\beta))}{\mathrm{gid}+\beta}},$$
$$g^{\frac{(\mathrm{gid}+\beta)}{r_i}})$$
$$= c \cdot e(g,g)^{y \prod_{\mathrm{gid}_x \in \alpha^2}(\mathrm{gid}_x+\beta)} / e(g,g)^{y \prod_{\mathrm{gid}_x \in \alpha^2}(\mathrm{gid}_x+\beta)}$$
$$= c.$$

We want to point out a couple of details:

a) Every c' is tagged with an accumulator external identifier (described in $\mathrm{ReEnc}(\cdot)$). c' is selected in the calling process based on a given external identifier. The details will be specified in the system construction section.

b) Although some notations, such as y versus y_i, c' versus c'_i , and gid versus gid_i, look similar, the notation without subscript specifies individual entities, and the notation with subscript indicates members inside an entity set. Therefore, they may refer to the same value in algorithms.

(2) Return c. Note that an incorrect or random string or value is returned if $\mathrm{wsk}_{\mathrm{gid}}$ cannot decrypt c'.

8. $\mathrm{Dec}_{\mathrm{ABE}}(c, \cdot)$: This is the ABE decryption algorithm. It takes the ciphertext c and the remaining parameters denoted as '$\cdot$' as the inputs. A user's ABE private key $\mathrm{usk}_{\mathrm{ABE}}$ is assumed to be a part of '$\cdot$'. It outputs m if $\mathrm{usk}_{\mathrm{ABE}}$ can decrypt c. Otherwise, the output depends on the output of $\mathrm{Dec}_{\mathrm{ABE}}$.

5.2 User revocation via ciphertext re-encryption (UR-CRE)

Our first approach is to achieve user revocation through ciphertext re-encryption.

5.2.1 Trust model

We assume that the storage system consists of four entities (Fig. 1):

1. AA: This is the AA of an ABE scheme. It is trusted to generate ABE private keys (or attribute key shares) for users, publish the ABE public key $\mathrm{pk}_{\mathrm{ABE}}$, and protect the ABE master secret $\mathrm{mk}_{\mathrm{ABE}}$.

2. DO: The DO is a trusted party who is responsible for data and user privacy protection. DO initializes the system, defines data access policy, and centrally manages user access rights to the encrypted data in cloud storage.

3. CSP: CSPs provide cloud storage for a DO to store data and fulfill user requests. It is considered to be untrusted by DOs.

4. Users: Users can decrypt data only if they are eligible and have attributes complying with the access control policy of an ABE scheme. Users are considered to be untrusted by DOs.

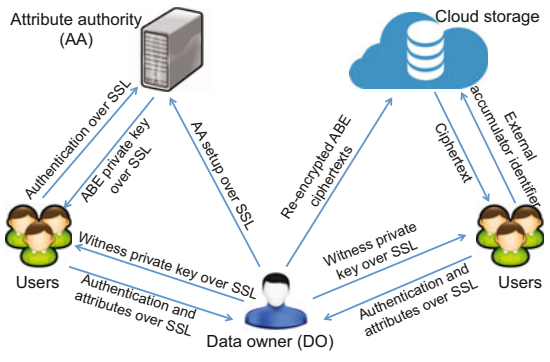

Fig. 1 Trust model of user revocation via ciphertext re-encryption for attribute-based encryption (ABE) in cloud storage

We assume that, where necessary, communications among entities in our system are protected via suitable secure communication mechanisms, such as

using SSL/TLS. We also assume that CSPs and users do not collude with each other during the data retrieval process.

We assume that each user is assigned with a unique gid that is not linked to their true identity. To simplify the description of our system, we do not elaborate the user verification process. There are multiple ways to verify whether a gid belongs to a user.

5.2.2 System description

Fig. 2 shows the interactions among a user, user groups, a DO, an AA, and a CSP. The ABE scheme might have multiple attribute authorities. However, because ABE schemes are treated as black boxes, the AA in our description is the generic.

The system works as follows:

1. System setup: This is a private process run by a DO. The process takes in a security parameter 1^n and initializes system parameters, such as an ABE scheme and a DA. Then it adds eligible (non-revoked) users to accumulators, issues and sends witness private keys to users, encrypts data using ABE, and re-encrypts ABE ciphertext c for each accumulator.

The outputs of the process are the following:

(1) The ABE public key $\mathrm{pk}_{\mathrm{ABE}}$ and master secret $\mathrm{mk}_{\mathrm{ABE}}$ are sent to the AA. $\mathrm{pk}_{\mathrm{ABE}}$ is public. $\mathrm{mk}_{\mathrm{ABE}}$ needs to be kept secret.

(2) A witness private key $\mathrm{wsk}_{\mathrm{gid}}$ is sent to each user and kept secret.

(3) The re-encrypted ciphertext C' is sent to the CSP and kept public.

The detailed steps are as follows:

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

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

(3) The DO creates accumulators and issues witness private keys:

a) Let U contain the existing non-revoked gids.

b) Let WitnessSet = {} be the temporary data structure for the returned witnesses of an accumulator.

c) For each $\mathrm{gid}_i \in U$, the DO adds gid_i to an accumulator and generates its witness:

c1) Call UserManager(gid_i, 'Add', Φ) and obtain $\{\omega_\alpha, \alpha\}$ as the output;

c2) Check if α has been added to WitnessSet or not by using the internal index α^3: if α exists, then replace the existing $\{\omega_\alpha, \alpha\}$ with the new one; else, WitnessSet = WitnessSet + $\{\omega_\alpha, \alpha\}$.

Note: This is a one-time bulk process to add users into accumulators. Although each gid is newly added to the system, it could be added to an existing accumulator. Therefore, the witnesses of the accumulator need to be updated. An accumulator can be updated $k-1$ times until the number of gids in the accumulator reaches the maximum number of allowed elements, k. Because this internal computation takes place on the DO side, no user is aware of it until the next step. This operation can be optimized in the actual implementation.

d) The DO issues the witness private keys as follows:

For each $\{\omega_\alpha, \alpha\} \in$ WitnessSet:

d1) Call WitnessKeyGen(ω_α, α) and obtain witKeys;

d2) For each $\mathrm{gid}_i \in$ witKeys, send the witness key $\mathrm{wsk}_{\mathrm{gid}_i}$ to user gid_i.

(4) The DO calls $\mathrm{Enc}_{\mathrm{ABE}}(m, \cdot)$ to encrypt the message m and obtains ciphertext c.

(5) The DO calls ReEnc(c, Φ) to re-encrypt c and obtains $C' = \{\{c'_i, g^{y_i}\}\}_{1 \le i \le |\Phi|}$.

(6) Finally, the DO sends C' to the CSP.

2. A new user (gid) requests a witness private key. This process is privately run by the DO. If the user is eligible, the process will add the user to an accumulator. All users in the accumulator are updated with new witness private keys. The ABE ciphertext is re-encrypted.

The outputs of the process are the following:

(1) New witness private keys are sent to users of the updated accumulator and kept secret.

(2) The updated ciphertext C' is sent to the CSP and kept public.

The detailed steps are as follows:

(1) The user sends a gid to the DO.

(2) The DO authenticates and validates the user:

a) If the user is eligible, the DO does the following:

a1) Calls UserManager(gid, 'Add', Φ) to add gid and obtains $\{\omega_\alpha, \alpha\}$ as the output.

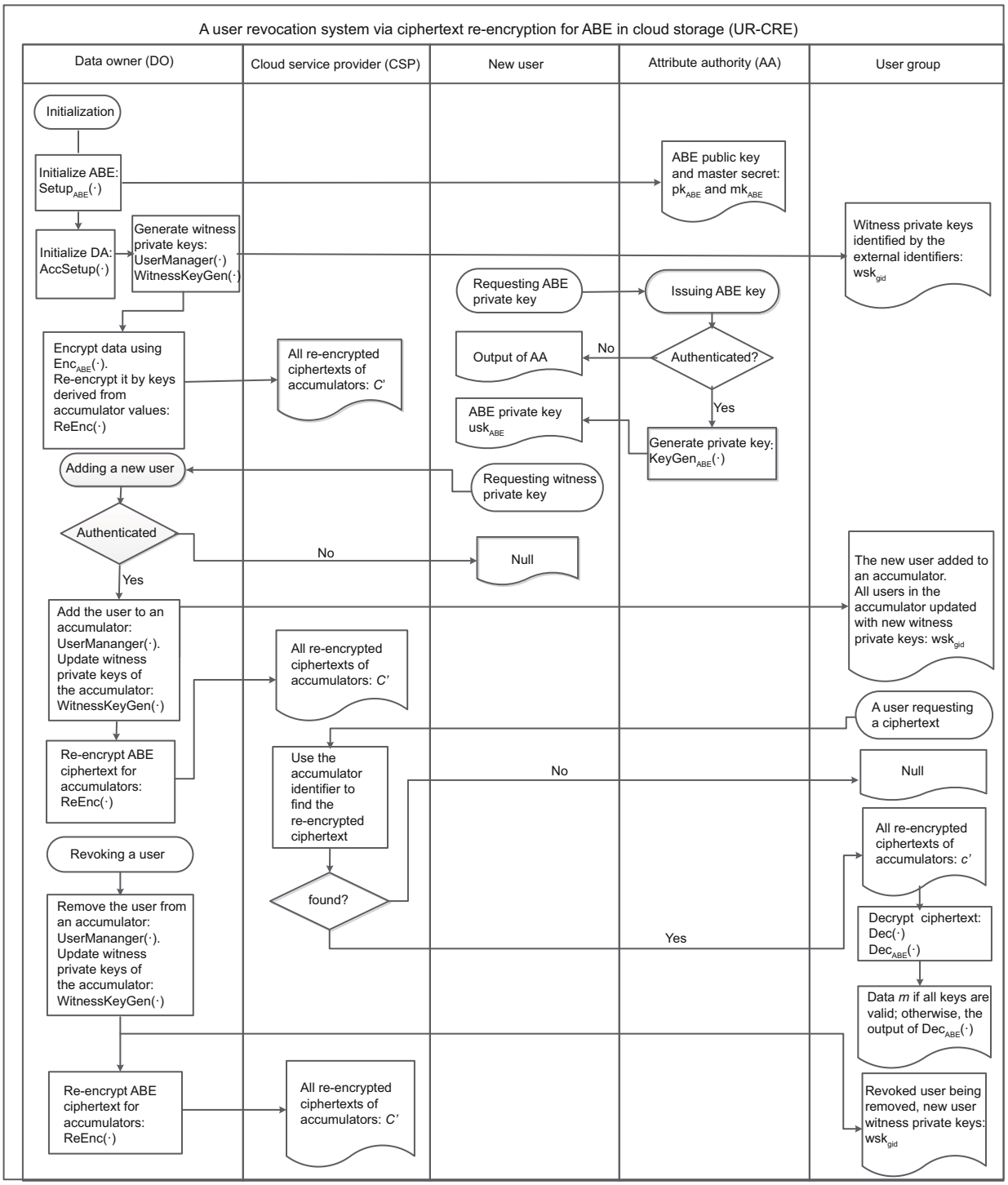

Fig. 2 Interaction diagram of user revocation via ciphertext re-encryption for attribute-based encryption (ABE) in cloud storage

a2) Calls WitnessKeyGen(ω_α, α) and obtains witKeys.

a3) For each $\text{gid}_i \in$ witKeys, sends the witness key $\text{wsk}_{\text{gid}_i}$ to user gid_i.

a4) Calls ReEnc(c, Φ) to re-encrypt c for each $\alpha_i \in \Phi$ and obtains $C' = \{\{c_i', g^{y_i}\}\}_{1 \le i \le |\Phi|}$.

We re-encrypt the ABE ciphertext c for every accumulator in Φ although only one accumulator has

been changed. In this way, the CSP simply replaces the previous C' with the newly received C'.

a5) Send C' to the CSP.

b) Otherwise, the DO returns $\perp$ to the user.

3. A new user (gid) requests an ABE private key. This is a private process run by the AA. The output of the process is one of the following:

(1) An ABE private key usk_{ABE} is sent to the user if the user is authenticated.

(2) Otherwise, the output of the AA could be $\perp$, as an example.

The detailed steps are as follows:

(1) The user contacts the AA to request his/her ABE private key.

(2) The AA authenticates the user:

a) If the user can be authenticated, then the AA does the following:

a1) Calls $\text{KeyGen}_{\text{ABE}}(\cdot)$ and generates usk_{ABE}.

a2) Sends usk_{ABE} to the user.

b) Otherwise, the AA returns its output.

4. A user (gid) retrieves a ciphertext from the CSP. This process is publicly run by the CSP and a user. The user sends the CSP the external identifier included in the user's witness private key. The CSP locates the re-encrypted ciphertext tagged with the external identifier if the re-encrypted ciphertext exists. Upon receiving the re-encrypted ciphertext, the user decrypts it by calling Dec_{ABE} to obtain the ABE ciphertext and Dec_{ABE} to decrypt the data.

The output of the process is one of the following:

(1) $\perp$, if the re-encrypted ciphertext is not found;

(2) otherwise, either data m, if the user is not revoked and has a valid ABE private key, or the output of Dec_{ABE}.

The detailed steps are as follows:

(1) User gid sends the external accumulator identifier $\text{wsk}_{\text{gid}}^{3}$ (g^{y}) to the CSP.

(2) The CSP returns the ciphertext c' identified by g^{y} if c' exists; otherwise, $\perp$ is returned.

(3) If the value returned is not $\perp$:

a) The user calls $\text{Dec}(c', \text{wsk}_{\text{gid}})$:

a1) If wsk_{gid} is a valid witness key of g^{y}, c is returned;

a2) Otherwise, a random string is returned.

b) The user calls $\text{Dec}_{\text{ABE}}(c, \cdot)$:

b1) If c is in a correct form and usk_{ABE} can decrypt it, m is returned;

b2) Otherwise, the output of Dec_{ABE} is returned.

5. The DO revokes a user (gid). This is a private process run by the DO. The DO removes the user from the user's accumulator, updates the remaining users with new witness private keys, and re-encrypts ABE ciphertext c to C'.

The outputs of the process are the following:

(1) New witness private keys are sent to the users of the updated accumulator and kept secret.

(2) The re-encrypted ciphertext C' is sent to the CSP and kept public.

The detailed steps are as follows:

(1) The DO calls UserManager(gid, 'Delete', Φ) to remove the gid and obtains the output $\{\omega_{\alpha}, \alpha\}$.

(2) If $\alpha^{2} \neq \{\}$, the DO does the following:

a) Calls $\text{WitnessKeyGen}(\omega_{\alpha}, \alpha)$ to generate witness private keys witKeys;

b) For $\text{gid}_i \in \alpha^{2}$, sends $\text{wsk}_{\text{gid}_i}$ to user gid_i.

(3) The DO calls $\text{ReEnc}(c, \Phi)$ to re-encrypt c and obtains $C' = \{\{c_i', g^{y_i}\}\}_{1 \leq i \leq |\Phi|}$ as the output.

(4) The DO sends C' to the CSP for the update.

5.2.3 Security analysis

The ciphertext re-encryption system (UR-CRE) generically enables an anonymous user revocation for ABE in untrusted cloud storage environments. The system is built on top of an ABE scheme and leverages a dynamic accumulator (DA) based on the scheme proposed by Au et al. (2009). Therefore, the fundamental security of the system relies on the security of the selected ABE scheme and the DA. We assume that:

1. The ABE scheme is at least CPA-secure.

2. The DA scheme is secure against the forgeability of witnesses and provides anonymity protection for elements aggregated into accumulators.

The capability of revocation is aimed to equip an ABE scheme with two additional security features:

1. A revoked user cannot immediately access ABE ciphertext after the revocation. However, this protection does not apply to the ciphertext that has been requested or possibly kept by the user locally before the revocation.

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

The two security features require the system to have the following:

1. Dynamic user revocation: Revoked users who have valid ABE private keys are immediately prevented from accessing ABE ciphertext.

2. Ciphertext indistinguishability: The re-encrypted ABE ciphertext should remain indistinguishable against eavesdropping attacks.

3. Unforgeability: Users should not be able to forge their witness private keys to decrypt the re-encrypted ABE ciphertexts.

4. Anonymity: Users remain anonymous to CSPs. CSPs are not required for any user management or administration to fulfill data retrieval requests. In addition, the privacy and integrity of ABE access policies (expressed by user attributes) are protected by the re-encryption.

The system security is analyzed as follows:

1. Security of dynamic user revocation

Dynamic user revocation is realized by preventing revoked users from accessing ABE ciphertexts.

Each user has two types of private keys: a witness private key for decrypting a re-encrypted ABE ciphertext and an ABE private key for decrypting an ABE ciphertext. Once a user is revoked, the AA does not re-issue ABE private keys. Instead, the DO takes the following actions to invalidate the user's witness private key:

(1) First, the DO removes the user from the user's accumulator. Assume that accumulator α_i is the one containing the user. After the user is removed, α_i is updated with the new aggregate value ($\alpha_i^1 = v_i'$). The external identifier g^{y_i} is re-generated as $g^{y_i'}$, where y_i' is randomly selected from Z_p. All accumulator data structures, α and Φ, are kept secret by the DO. y_i and y_i' are not provided to users and CSPs. Only the external identifiers (in the form of g^y) are public and given to users and the CSPs.

(2) Then, the DO generates new witness private keys to the remaining users in α_i. As being described in the algorithm WitnessKeyGen, each user's witness private key is derived from the user's new witness of α_i by randomizing it with y_i' and r', which is also randomly and uniquely selected from Z_p to each user every time.

Because the revoked user is removed from α_i and does not belong to any other accumulator, the user is not updated with a new witness key. The user cannot forge a witness of any accumulator based on the unforgeability of DA. All numbers y_i are privately kept by the DO. Based on the DLP assumption, the revoked user cannot discover y_i' even though the user can obtain $g^{y_i'}$. Without the needed components, v_i' (α_i^1) and y_i', the revoked user cannot forge a valid witness private key once being revoked.

(3) The DO re-encrypts the ABE ciphertext with the new accumulator value v_i' and the external identifier y_i': $c_i' = ce(g^{y_i'}, v_i')$. Although the revoked user cannot forge the new witness of an accumulator, the user can try to forge the new aggregate value. Assume that this user is the only user who has been removed from the accumulator. Because the DA's system parameters, such as accumulator trapdoor β and function A_k, are kept secret by the DO, the user needs to obtain the previous v_i from his/her witness private key obtained before his/her revocation. Assume the witness private key is $\mathrm{wsk}_{\mathrm{gid}} = \{\mathrm{wsk}_{\mathrm{gid}}^1 = (w_{\mathrm{gid}}^1)^r = g^{\frac{ry_i(\prod_{\mathrm{gid}_x \in \alpha_i^2}(\mathrm{gid}_x+\beta))}{\mathrm{gid}+\beta}}$, $\mathrm{wsk}_{\mathrm{gid}}^2 = (w_{\mathrm{gid}})^{1/r} = g^{\frac{(\mathrm{gid}+\beta)}{r}}$, $\mathrm{wsk}_{\mathrm{gid}}^3 = (g)^{\alpha^4} = g^{y_i}\}$. Under the DLP assumption, the user cannot obtain $\prod_{\mathrm{gid}_x \in \alpha_i^2}(\mathrm{gid}_x+\beta)$ to compute $v_i = g^{\prod_{\mathrm{gid}_x \in \alpha_i^2}(\mathrm{gid}_x+\beta)}$. Therefore, the user cannot forge v_i' which is equal to $v_i^{\frac{1}{\mathrm{gid}+\beta}}$.

Based on the above analysis, the user cannot decrypt any re-encrypted ABE ciphertext, because without the witness private key, the original ABE private key is insufficient for decryption.

It is possible that a non-revoked user has a valid witness private key, but the attributes fail to comply with the ABE's access policy. This might happen because the witness private keys are issued and used separately from the keys used in ABE. In this case, the security of the ABE scheme should prevent the user from decrypting the ciphertext.

2. Security of ciphertext indistinguishability

The system should resist eavesdropping adversaries.

The communication channels between the DO and the CSPs, or users and CSPs, might be unprotected. An adversary can eavesdrop on ciphertexts sent in unsecured channels. Assume that an adversary can obtain the ABE public (encryption) key and the re-encrypted ciphertexts from unencrypted channels. If the adapted ABE is CPA-secure, ABE ciphertext is indistinguishable against eavesdropping attacks. Because the re-encryption keys (in the form of $e(g^y, v)$) are randomized intuitively (by uniformly selected y) for each accumulator and changed for every subsequent ciphertext re-encryption, the re-encrypted ciphertext is also indistinguishable.

3. Security of unforgeability

The system prevents a revoked user from forging the new ciphertext re-encryption key based on the witness private key that the user acquired before the revocation. This security is based on the membership unforgeability of the DA (scheme proposed by Au et al. (2009)) and the DLP assumption. We use the following scenario to informally analyze this. Assume that a user (A) with gid is aggregated in α_i.

(1) At the beginning, user A is given his/her witness key of α_i as

$$\mathrm{wsk}_{\mathrm{gid}}=\left\{\begin{aligned}&\mathrm{wsk}^1_{\mathrm{gid}}=(w^1_{\mathrm{gid}})^r=g^{\frac{ry_i(\prod_{\mathrm{gid}_x\in\alpha^2_i}(\mathrm{gid}_x+\beta))}{\mathrm{gid}+\beta}},\\&\mathrm{wsk}^2_{\mathrm{gid}}=(w_{\mathrm{gid}})^{1/r}=g^{\frac{(\mathrm{gid}+\beta)}{r}},\\&\mathrm{wsk}^3_{\mathrm{gid}}=(g)^{\alpha^4}=g^{y_i}\end{aligned}\right\}.$$

(2) When A is revoked, the DO removes gid from α_i, and y_i' is randomly selected. The accumulator α_i has the new external identifier $g^{y_i'}$ and the new aggregate value v_i'. Then DO refreshes the witness private keys of the remaining members in α_i^2 (which does not include the revoked gid any more) .

(3) A is free to retrieve any number of re-encrypted ciphertexts from CSPs within polynomial time. Because A is not a member of any accumulator, A cannot forge any witness of an accumulator based on the security of DA. A cannot extract v_i from his/her previous witness private key either, as explained in Section 5.2.3. Therefore, A cannot reconstruct any re-encryption key of the ciphertext, even though $g^{y_i'}$ might be obtained by A.

4. Security of anonymity

The anonymity protection for users is anonymous to CSPs. User identities are protected in the following ways:

(1) Each user is assigned a pseudo-identifier gid that is unique and is not linked to a real identity.

(2) An accumulator's aggregate value (v) does not reveal any aggregated gids based on the security of DA (Au et al., 2009).

(3) CSPs are provided with any accumulator value and information other than re-encrypted ciphertexts.

Based on the above analysis, we can conclude that the proposed user revocation system has met the security goals.

Although collusion protection is one of the security requirements of ABE schemes, we do not consider it in our proposed system. This is because ABE schemes use one master secret to derive user private keys. An encryption and decryption key can include multiple parts, each related to a particular attribute. Because attributes are shared by users, users without all valid key parts or attributes can pool the key parts to construct a valid private key. Therefore, collusion prevention is a critical security requirement of ABE. However, this type of collusion does not apply to our proposed user revocation system, which builds an additional layer above the ABE scheme. There is no attribute sharing between the ABE scheme and the revocation system.

5.3 User revocation via cloud storage providers

Although using the ciphertext re-encryption approach, UR-CRE enables user revocation at the data item level and requires no user management on CSPs, and it introduces extra overhead in the following aspects:

1. It requires CSPs to store multiple re-encryption copies of an ABE ciphertext.

2. It requires a DO to re-encrypt the ABE ciphertext for an accumulator whenever it adds or removes a user.

To eliminate the above overhead, the second generic user revocation system, UR-CSP, leverages CSPs to anonymously identify revoked users during the ciphertext retrieval process. In this way, one copy of the ABE ciphertext needs to be stored. Only non-revoked users can obtain the ABE ciphertext from CSPs.

5.3.1 Trust model

The trust model still has four entities. The only difference is that CSPs become semi-trusted instead of being completely untrusted. The CSPs are trusted to perform user management in regard to anonymously identifying a user's eligibility to access an ABE ciphertext.

CSPs store ABE ciphertexts, validate user access rights, and fulfill data retrieval requests. CSPs are semi-trusted to execute the assigned tasks but might be curious to know the data and user identities.

The communication channels between entities in our framework are assumed to be secure, as shown

in Fig. 3. We also assume that CSPs and users do not collude with each other.

Each user is assigned a global unique identifier gid that is not linked to the user's true identity. We do not elaborate the verification process of identifying whether a gid belongs to a user or not. There are multiple ways of verifying a user's identity against the given gid, such as certifications.

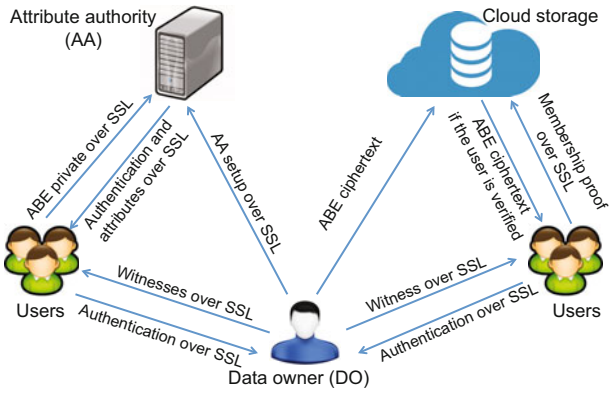

Fig. 3 Trust model of user revocation via cloud service providers for attribute-based encryption (ABE)

5.3.2 System description

This second user revocation system, UR-CSP, still uses DA to manage user access rights as in UR-CRE, except that users are given witnesses instead of witness private keys. Each witness is used to prove a user (gid) being aggregated to an accumulator. If CSPs can validate a user's claim to an accumulator, the requested ABE ciphertext is returned.

The external accumulator identifiers are not tagged to any user witness and accumulators' aggregate values. Instead, α^4 (the randomly selected number when α was created or updated) is used to randomize the witnesses that are sent to users and aggregate values (α^1) that are sent to the CSP. In this way, the aggregate values and witnesses obtain further protection from CSPs and users.

One way for a user to attest the membership of an accumulator is to directly send his/her witness to a CSP. Although each witness does not directly link to a user's identity, it does identify the individual user and the user's access privilege to a ciphertext. Because the CSP is not fully trusted, directly sharing a witness with the CSP could have potential risks, such as impersonation attacks launched by the malicious CSP insiders. Therefore, the witnesses are considered to be private and kept secret by users.

Inspired by zero knowledge proofs (Camenisch and Lysyanskaya, 2002), our approach is to make users compute witness proofs. Each time a user requests the ABE ciphertext, the user computes a witness proof using his/her witness. The witness proof is randomized each time, even on the same witness. The CSP can validate the proof against the accumulators' aggregate values provided by the DO. Fig. 4 shows the interactions between different parties in the system.

The details of the system construction are as follows:

1. System setup: This is a private process run by DO. It initializes the system by setting up system parameters, an ABE scheme, and a DA. Then it adds non-revoked users to accumulators, issues and sends witnesses, encrypts data using ABE, randomizes accumulators' aggregate values, and sends the ABE ciphertexts and randomized aggregate values to CSPs.

The outputs of the process are the following:

(1) ABE $\mathrm{pk}_{\mathrm{ABE}}$ and $\mathrm{mk}_{\mathrm{ABE}}$ are sent to the AA. $\mathrm{pk}_{\mathrm{ABE}}$ is public. $\mathrm{mk}_{\mathrm{ABE}}$ is kept secret.

(2) Witnesses w_{gid} is sent to eligible users to be kept secret.

(3) The ABE ciphertexts c and Λ (the container described below for storing randomized aggregate values) are sent to the CSP. c and Λ can be public.

The detailed steps are as follows:

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

(2) The DO provides the AA with the public key $\mathrm{pk}_{\mathrm{ABE}}$ and master secret $\mathrm{mk}_{\mathrm{ABE}}$.

(3) Let U contain the existing non-revoked gids.

a) Let WitnessSet = {} be the container to temporarily store the returned witnesses of the accumulators.

b) For each $\mathrm{gid}_i \in U$, DO proceeds as follows:

b1) Calls UserManager(gid_i, 'Add', Φ) to add gid_i to an accumulator and obtains $\{\omega_\alpha, \alpha\}$ as the output;

b2) Checks whether α has been added to WitnessSet or not by using the internal index α^3. If α exists, removes the existing $\{\omega_\alpha, \alpha\}$.

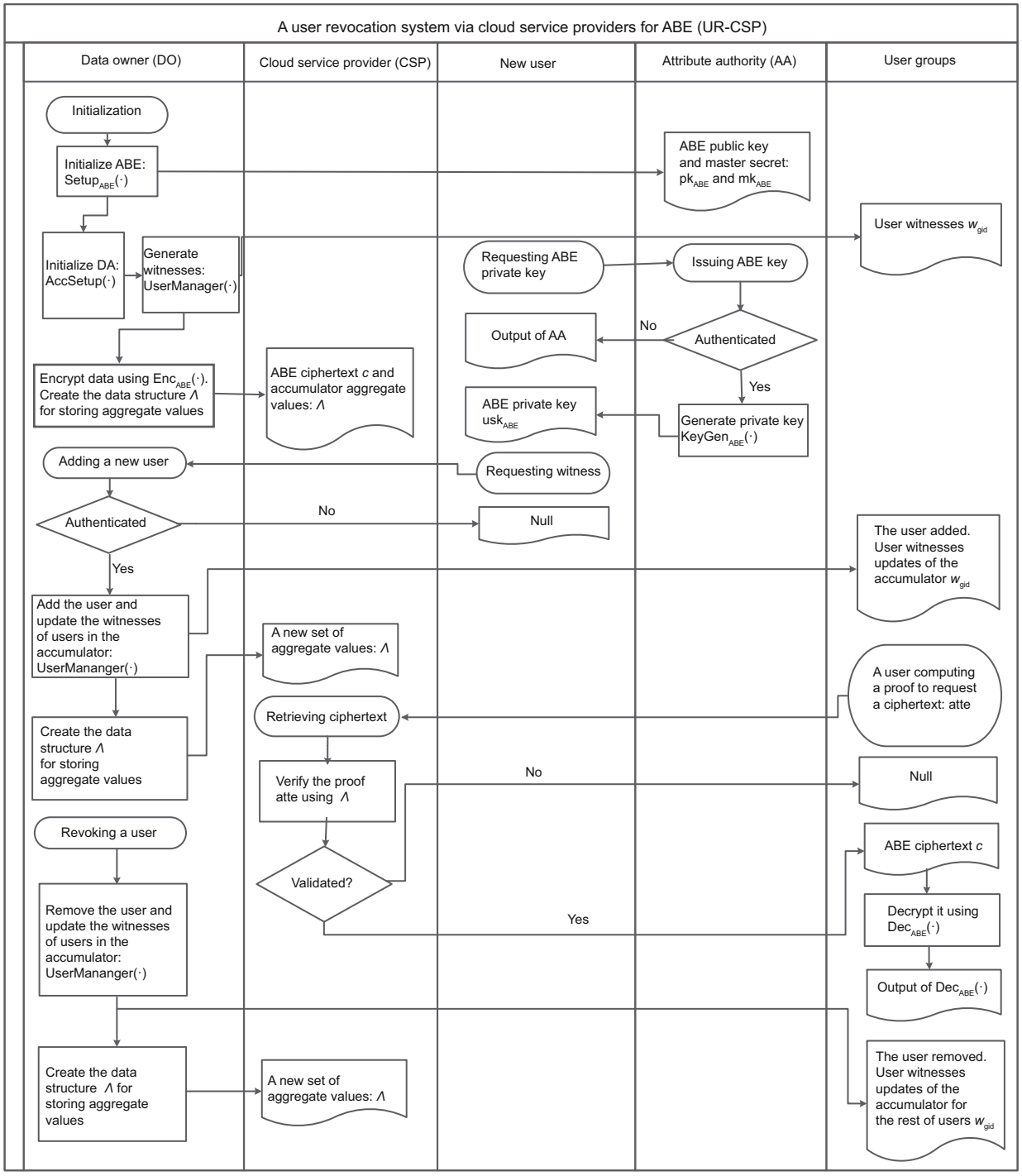

Fig. 4 Interaction diagram of user revocation via cloud service providers for attribute-based encryption (ABE)

b3) Updates WitnessSet: WitnessSet $\leftarrow$ WitnessSet + $\{\omega_\alpha, \alpha\}$.

Note: Again, this is a one-time bulk process for adding all users into accumulators.

c) For each $\{\omega_\alpha, \alpha\} \in$ WitnessSet:

For each $\mathrm{gid}_i \in \alpha^2$, the DO re-computes $w_{\mathrm{gid}_i} = \{(w^1_{\mathrm{gid}_i})^{\alpha^4} = (w^1_{\mathrm{gid}})^y, w^2_{\mathrm{gid}_i}\}$ and sends w_{gid_i} to user gid_i.

(4) The DO calls $\mathrm{Enc}_{\mathrm{ABE}}(m, \cdot)$ to encrypt the message m and obtains ABE ciphertext c.

(5) Let Λ be the container to store all the current aggregate values.

For each accumulator $\alpha_i \in \Phi$: The DO adds v_i to $\Lambda = \Lambda + \{v_i^{y_i}\}$, where v_i is α_i^1 and y_i is α_i^4.

We use y_i to randomize aggregate values.

(6) The DO sends $\{\Lambda, c\}$ to the CSP.

2. The DO adds a new user (gid) to the system. This process is privately run by the DO. It adds the user to an accumulator and updates all users in the accumulator with new witnesses. Then it updates the CSP with the new randomized aggregate values.

The outputs of the process are as follows:

(1) New witnesses of the updated accumulator are sent to users to be kept secret.

(2) Updated Λ is sent to the CSP, and it can be public.

The detailed steps are as follows:

(1) The DO validates and authenticates the user gid.

(2) If the user is eligible:

a) DO calls UserManager(gid, 'Add', Φ) to add gid to an accumulator and has $\{\omega_\alpha, \alpha\}$ returned.

b) For each $\text{gid}_i \in \alpha^2$, the DO obtains y from α^4, re-computes $w_{\text{gid}_i} = \{(w^1_{\text{gid}_i})^y, w^2_{\text{gid}_i}\}$, and sends the witness w_{gid_i} to the user.

c) Let $\Lambda = \{\}$.

d) For each $\alpha_i \in \Phi$, DO adds v_i to Λ: $\Lambda = \Lambda + \{v_i^{y_i}\}$, where v_i is α_i^1 and y_i is α_i^4.

Note: The updated α has been included in Φ. Therefore, all the aggregate values are currently in Φ. Although only one accumulator is updated, Λ is re-generated for the CSP to simply replace the previous one completely.

e) The DO sends Λ to the CSP to replace the previous one.

(3) If the user is not eligible, the DO returns $\perp$.

3. A new user (gid) requests an ABE private key. This process is the same as described in the UR-CRE.

The output of the process is one of the following:

(1) The ABE private key usk_{ABE} is kept secret by the user if the user is authenticated.

(2) Otherwise, the output is the output of the AA, which could be $\perp$ as an example.

The detailed steps are as follows:

(1) The user contacts the AA with his/her attributes.

(2) The AA verifies the user and his/her attributes.

(3) If the user is successfully verified, the AA calls $\text{KeyGen}_{\text{ABE}}(\cdot)$ and sends the user usk_{ABE}.

(4) Otherwise, the AA outputs its output.

5. A user gid requests an ABE ciphertext. This process consists of the following communications between the CSP and the user:

(1) The user computes a witness proof and sends it to the CSP.

(2) Then, upon obtaining the request and the proof, the CSP checks the proof against the accumulator's aggregate values.

(3) If the proof can be verified, the CSP sends ciphertext c to the user; otherwise, the CSP returns $\perp$.

(4) If the user obtains c back, he/she calls Dec_{ABE} to decrypt the data.

The output of this process is one of the following:

(1) Data m, if the user can prove his/her membership and has a valid usk_{ABE}.

(2) Output of Dec_{ABE}, if the user can prove his/her membership but does not have a valid usk_{ABE}.

(3) $\perp$, if the user cannot prove his/her membership of an accumulator.

The detailed steps are as follows:

(1) The user gid computes a proof as follows:

a) Randomly select γ uniformly from Z_p and compute:

$$\begin{aligned}\text{atte} &= e\left(w^1_{\text{gid}}, (w^2_{\text{gid}})^\gamma\right) = e\left(v^{\frac{y}{\text{gid}+\beta}}, g^{\gamma(\text{gid}+\beta)}\right)\\ &= e\left(g^{\prod_{\text{gid}_x \in \alpha^2}(\text{gid}_x+\beta)}, g^{\gamma y}\right)\\ &= e\,(g,g)^{ry\prod_{\text{gid}_x \in \alpha^2}(\text{gid}_x+\beta)}.\end{aligned}$$

b) Compute g^γ.

(2) The user gid sends {atte, g^γ} to the CSP.

(3) The CSP validates atte:

a) For each $\{v_i^{y_i}\} \in \Lambda$:

a1) Compute

$e(v_i^{y_i}, g^\gamma) = e(v_i, g^{y_i\gamma}) = e(g, g)^{ry_i\prod_{\text{gid}_x \in \alpha_i^2}(\text{gid}_x+\beta)}$.

a2) Check whether

$$\text{atte} \stackrel{?}{=} e(g, g)^{ry_i\prod_{\text{gid}_x \in \alpha_i^2}(\text{gid}_x+\beta)}.$$

b) If one match is found, ABE ciphertext c is returned.

c) Otherwise, $\perp$ is returned.

(4) If c is returned, the user calls $\text{Dec}_{\text{ABE}}(c, \cdot)$ to obtain m if sk_{ABE} can decrypt c.

(5) Otherwise, the output depends on the output of $\text{Dec}_{\text{ABE}}(c, \cdot)$.

5. The DO revokes a user gid. This process is privately run by the DO. It removes the user from an accumulator and updates the remaining users with

new witnesses. The CSP is also provided with the updates.

The outputs of the process are the following:

(1) New witnesses are sent users to be kept secret, if the updated accumulator α has the remaining users.

(2) Updated Λ is sent to the CSP and can be public.

The detailed steps are as follows:

(1) The DO calls UserManager(gid, 'Delete', Φ) to remove gid from an accumulator and has $\{\omega_\alpha, \alpha\}$ returned.

(2) If $\alpha^2 \neq \{\}$:

For each $\mathrm{gid}_i \in \alpha^2$, the DO obtains the random number y from α^4, re-computes $w_{\mathrm{gid}_i} = \{(w^1_{\mathrm{gid}_i})^y, w^2_{\mathrm{gid}_i}\}$, and sends the witness w_{gid_i} to the user gid_i.

(3) Let $\Lambda = \{\}$.

(4) For each $\alpha_i \in \Phi$, the DO computes $v_i^{y_i}$ and adds it to Λ: $\Lambda = \Lambda + \{v_i^{y_i}\}$, where v_i is α_i^1 and y_i is α_i^4.

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

5.3.3 Security analysis

Because the UR-CSP is still built on top of an ABE scheme and leverages a DA to achieve dynamic and anonymous user revocation, the fundamental security of UR-CSP still relies on the security of ABE and the DA. We assume that:

1. The selected ABE scheme is at least CPA-secure.

2. The DA, which is built out of the scheme proposed by Au et al. (2009), is secure against witness forgeability and provides anonymity of elements aggregated into accumulators.

UR-CSP aims to achieve similar goals to the previous system UR-CRE:

(1) A revoked user cannot decrypt ABE ciphertexts after the revocation.

(2) CSPs can anonymously identify revoked users during the ciphertext retrieval process.

These goals require the same security as for UR-CRE:

(1) Dynamic user revocation: Revoked users having valid ABE private keys are prevented from accessing any ABE ciphertext after revocation.

(2) Ciphertext indistinguishability: The ABE ciphertext should remain indistinguishable against eavesdropping attacks.

(3) Unforgeability: Users should not be able to forge their witnesses to prove the membership of an accumulator.

(4) Anonymity: Users remain anonymous to CSPs in any user identification and data retrieval request.

The detailed analysis is as follows:

1. Security of dynamic user revocation

Assume a revoked user still holds a valid ABE private key for decryption. Then the system prevents the user from accessing the ABE ciphertext as follows:

(1) When the user (gid) is revoked, the DO removes the gid from its accumulator. Supposing that α is the accumulator, the external randomizing number α^4 is re-selected and the aggregate value α^1 is recalculated. Let y' and v' denote the updated values of α^4 and α^1, respectively. Each of the remaining users $\mathrm{gid}_i \in \alpha^2$ receives a new witness w'_{gid_i} randomized by y':

$$w'_{\mathrm{gid}_i} = \left\{ w'^1_{\mathrm{gid}_i} = v'^{\frac{y'}{\mathrm{gid}_i+\beta}}, w'^2_{\mathrm{gid}_i} = g^{\mathrm{gid}_i+\beta} \right\}.$$

The CSP is also updated with a new Λ (containing $(v')^{y'}$).

(2) The revoked user (gid) is not given any new or updated witness. Suppose that the revoked user has the 'old' witness and is denoted as follows: $w_{\mathrm{gid}} = \left\{ w^1_{\mathrm{gid}} = v^{\frac{y}{\mathrm{gid}_i+\beta}} = g^{\frac{y(\prod_{\mathrm{gid}_i\in\alpha}(\mathrm{gid}_i+\beta))}{\mathrm{gid}+\beta}}, \ w^2_{\mathrm{gid}} = g^{\mathrm{gid}+\beta} \right\}$, where y and v are the 'old' values.

a) Based on the DLP assumption, the user cannot extract $\frac{y\prod_{\mathrm{gid}_x\in\alpha}(\mathrm{gid}_x+\beta)}{\mathrm{gid}+\beta}$ and $\mathrm{gid}+\beta$ from his/her witness.

b) Suppose the user computes $e(w^1_{\mathrm{gid}}, w^2_{\mathrm{gid}})$ to obtain $e(g,g)^{y\prod_{\mathrm{gid}_x\subseteq\alpha}(\mathrm{gid}_x+\beta)}$. $y\prod_{\mathrm{gid}_x\in\prod\alpha}(\mathrm{gid}_x+\beta)$ cannot be extracted based on the DLP assumption, nor the original aggregate value v. Because the DA's system parameters and data structures are kept secret from users, the user is prevented from computing v'.

c) Based on the witness unforgebility of DA, the user cannot forge a witness of any accumulator.

d) Suppose that the user assumes him/her to be the only one removed from the accumulator. The user tries to forge a proof (atte) by randomly selecting γ' and uses only w^1_{gid} as atte $= e(w^1_{\mathrm{gid}}, (g)^{\gamma'}) = e(g,\ g)^{r'y\prod_{\mathrm{gid}_x\in\alpha}(\mathrm{gid}_x+\beta)}$. Because y has been changed to y', which is randomly selected and

kept secret by the DO, the atte cannot be correctly validated.

Based on the above analysis, the revoked user cannot generate a valid proof to the CSP. He/She is not given any ABE ciphertext after the revocation.

2. Security of ciphertext indistinguishability

Because UR-CSP uses an ABE scheme for data encryption, we assume that the security of ciphertext indistinguishability is provided by the ABE scheme selected by the DO.

3. Security of unforgeability

This security is based on the membership unforgeability of the scheme proposed by Au et al. (2009). Any user, whether revoked or not, cannot forge a valid witness of an accumulator that does not have the user's gid aggregated in it.

4. Security of anonymity

The user verification process does not need user identities or gids:

(1) Users use witnesses to compute a claim to prove their memberships of a particular accumulator:

$\text{atte} = e(w_{\text{gid}}^1, (w_{\text{gid}}^2)^{\gamma}) = e(v^{\frac{y}{\text{gid}+\beta}}, g^{\gamma(\text{gid}+\beta)}) = e(g, g)^{ry\prod_{\text{gid}_x\in\alpha^2}(\text{gid}_x+\beta)}$.

γ is randomly chosen each time; therefore, atte is differently formed even using the same witness.

(2) Using the given aggregate values $\Lambda = \{\{v_i^{y_i}\}_{1\le i\le|\Phi|}\}$, the CSP verifies a claim by checking the following:

For each $v_i^{y_i} \in \Lambda$:

a) Compute

$e(v_i^{y_i}, g^{\gamma}) = e(g, g)^{ry_i\prod_{\text{gid}_x\in\alpha_i^2}(\text{gid}_x+\beta)}$.

b) Check whether

$\text{atte} \stackrel{?}{=} e(g, g)^{ry_i\prod_{\text{gid}_x\in\alpha_i^2}(\text{gid}_x+\beta)}$.

None of the above steps and information will disclose user identities to the CSP. Thus, users remain anonymous to the CSP in the processes of user verification and data retrieval.

Based on the above analysis, we can conclude that the second proposed user revocation system, UR-CSP, has met the security requirements.

6 User revocation overhead analysis

We are not aware of any similar system that can generically work with any existing ABE scheme. Thus, we simply compare our two systems in this section (Fig. 5).

The user revocation systems add integration and management overheads. As we can see, the overheads come mainly from the following aspects:

1. accumulator management;

2. user witness or witness private key management;

3. ciphertext re-encryption and re-encryption updates to CSPs;

4. user membership construction and verification;

5. storage required for storing multiple copies of the re-encrypted ciphertext.

UR-CRE has overheads 1, 2, 3, and 5. Overheads 1, 2, and 3 are on the data owner side. CSPs are required only to provide extra storage space for multiple copies of the re-encrypted ABE ciphertexts (overhead 5). The data owner can be a potential bottleneck for updates and witness private key management. However, the performance and scalability can be greatly improved by delegating the witness private key updates and ciphertext re-encryption to a trusted party or servers. Transferring updates to users and CSPs can also be out-of-band.

UR-CSP has the overheads 1, 2, and 4. Overheads 1 and 2 are still on the data owner side, but the data owner does not need to re-encrypt ABE ciphertexts. The data owner can still be a bottleneck in managing accumulators and witness updates.

User revocation system	Data owner management	Storage required at a CSP	User witness or key refreshing	Data retrieval
UR-CRE	1. Accumulator management 2. User witness key management 3. ABE ciphertext re-encryptions and updates to CSPs	Multiple copies of ABE ciphertext	The witness keys and key updates provided by the data owner	The re-encrypted ciphertext sent to the user
UR-CSP	1. Accumulator management 2. User witness management 3. Accumulator value updates to CSPs	One copy of ABE ciphertext	The witnesses and witness updates provided by the data owner	1. User verification CSPs 2. ABE ciphertext sent to a verified user

Fig. 5 Overhead comparison of user revocation systems

Delegation can still be an implementation strategy to reduce the bottleneck. Only one copy of ABE ciphertext is stored at the CSP. However, ciphertext retrieval is a protocol of the user membership verification process (overhead 4). It requires a user to construct a proof for a CSP to validate the proof every time. CSPs are required to be semi-trusted to perform user membership verification. This trust is not required in UR-CRE.

Overall, the overheads of UR-CSP appear to be more evenly distributed compared to UR-CRE. It is slightly more scalable than UR-CRE, but UR-CRE can improve the scalability by delegating the tasks and computations.

7 Conclusions

In this paper, we have proposed two dynamic user revocation systems for ABE schemes. Our systems are generic and can be directly applied to any ABE scheme. We have built user privacy protection into the data retrieval process, making ABE schemes more suitable and practical for deployment in untrusted cloud storage systems.

References

Attrapadung N, Libert B, de Panafieu E, 2011. Expressive key-policy attribute-based encryption with constant-size ciphertexts. *LNCS*, 6571:90-108. https://doi.org/10.1007/978-3-642-19379-8_6

Attrapadung N, Herranz J, Laguillaumie F, et al., 2012. Attribute-based encryption schemes with constant-size ciphertexts. *Theor Comput Sci*, 422(9):15-38. https://doi.org/10.1016/j.tcs.2011.12.004

Au MH, Tsang PP, Susilo W, et al., 2009. Dynamic universal accumulators for DDH groups and their application to attribute-based anonymous credential systems. *LNCS*, 5473:295-308. https://doi.org/10.1007/978-3-642-00862-7_20

Benaloh J, de Mare M, 1993. One-way accumulators: a decentralized alternative to digital signatures. *LNCS*, 765:274-285. https://doi.org/10.1007/3-540-48285-7_24

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, Franklin M, 2003. Identity-based encryption from the Weil pairing. *SIAM J Comput*, 32(3):586-615. https://doi.org/10.1137/S0097539701398521

Brickell E, Camenisch J, Chen LQ, 2004. Direct anonymous attestation. Proc 11th ACM Conf on Computer and Communications Security, p.132-145. https://doi.org/10.1145/1030083.1030103

Camenisch J, Lysyanskaya A, 2002. Dynamic accumulators and application to efficient revocation of anonymous credentials. *LNCS*, 2442:61-76. https://doi.org/10.1007/3-540-45708-9_5

Canetti R, Halevi S, Katz J, 2004. Chosen-ciphertext security from identity-based encryption. *LNCS*, 3027:207-222. https://doi.org/10.1007/978-3-540-24676-3_13

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

Chase M, 2007. Multi-authority attribute based encryption. Proc 4th Conf on Theory of Cryptography, p.515-534. https://doi.org/10.1007/978-3-540-70936-7_28

Chase M, Chow SS, 2009. Improving privacy and security in multi-authority attribute-based encryption. Proc 16th ACM Conf on Computer and Communications Security, p.121-130. https://doi.org/10.1145/1653662.1653678

Chen C, Zhang ZF, Feng DG, 2011. Efficient ciphertext policy attribute-based encryption with constant-size ciphertext and constant computation-cost. *LNCS*, 6980:84-101. https://doi.org/10.1007/978-3-642-24316-5_8

Chen C, Chen J, Lim HW, et al., 2013. Fully secure attribute-based systems with short ciphertexts/signatures and threshold access structures. *LNCS*, 7779:50-67. https://doi.org/10.1007/978-3-642-36095-4_4

Cheung L, Newport C, 2007. Provably secure ciphertext policy ABE. Proc 14th ACM Conf on Computer and Communications Security, p.456-465. https://doi.org/10.1145/1315245.1315302

Chow R, Golle P, Jakobsson M, et al., 2009. Controlling data in the cloud: outsourcing computation without outsourcing control. Proc ACM Cloud Computing Security Workshop, p.85-90. https://doi.org/10.1145/1655008.1655020

Emura K, Miyaji A, Nomura A, et al., 2009. A ciphertext-policy attribute-based encryption scheme with constant ciphertext length. *LNCS*, 5451:13-23. https://doi.org/10.1007/978-3-642-00843-6_2

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

Goyal V, Pandey O, Sahai A, et al., 2006. Attribute-based encryption for fine-grained access control of encrypted data. Proc 13th ACM Conf on Computer and Communications Security, p.89-98. https://doi.org/10.1145/1180405.1180418

Goyal V, Jain A, Pandey O, et al., 2008. Bounded ciphertext policy attribute based encryption. *LNCS*, 5126:579-591. https://doi.org/10.1007/978-3-540-70583-3_47

Han JG, Susilo W, Mu Y, et al., 2012. Privacy-preserving decentralized key-policy attribute-based encryption. *IEEE Trans Parall Distrib Syst*, 23(11):2150-2162. https://doi.org/10.1109/TPDS.2012.50

Hayes B, 2008. Cloud computing. *Commun ACM*, 51(7):9-11. https://doi.org/10.1145/1342327.1342330

Herranz J, Laguillaumie F, Ràfols C, 2010. Constant size ciphertexts in threshold attribute-based encryption. *LNCS*, 6056:19-34. https://doi.org/10.1007/978-3-642-13013-7_2

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

Ibraimi L, Tang Q, Hartel P, et al., 2009. Efficient and provable secure ciphertext-policy attribute-based encryption schemes. *LNCS*, 5451:1-12. https://doi.org/10.1007/978-3-642-00843-6_1

Jahid S, Mittal P, Borisov N, 2011. Easier: encryption-based access control in social networks with efficient revocation. Proc 6th ACM Symp on Information, Computer and Communications Security, p.411-415. https://doi.org/10.1145/1966913.1966970

Junod P, Karlov A, 2010. An efficient public-key attribute-based broadcast encryption scheme allowing arbitrary access policies. Proc 10th Annual ACM Workshop on Digital Rights Management, p.13-24. https://doi.org/10.1145/1866870.1866875

Karchmer M, Wigderson A, 1993. On span programs. Proc 8th Annual Structure in Complexity Theory Conf, p.102-111. https://doi.org/10.1109/SCT.1993.336536

Lewko A, Waters B, 2011. Decentralizing attribute-based encryption. *LNCS*, 6632:568-588. https://doi.org/10.1007/978-3-642-20465-4_31

Lewko A, Sahai A, Waters B, 2010a. Revocation systems with very small private keys. IEEE Symp on Security and Privacy, p.273-285. https://doi.org/10.1109/SP.2010.23

Lewko A, Okamoto T, Sahai A, et al., 2010b. Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption. *LNCS*, 6110: 62-91. https://doi.org/10.1007/978-3-642-13190-5_4

Li J, Huang Q, Chen XF, et al., 2011. Multi-authority ciphertext-policy attribute-based encryption with accountability. Proc ACM Symp on Information, Computer and Communications Security, p.386-390. https://doi.org/10.1145/1966913.1966964

Lin H, Cao ZF, Liang XH, et al., 2008. Secure threshold multi authority attribute based encryption without a central authority. *LNCS*, 5365:426-436. https://doi.org/10.1007/978-3-540-89754-5_33

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

Nguyen L, 2005. Accumulators from bilinear pairings and applications. *LNCS*, 3376:275-292. https://doi.org/10.1007/978-3-540-30574-3_19

Okamoto T, Takashima K, 2010. Fully secure functional encryption with general relations from the decisional linear assumption. *LNCS*, 6223:191-208. https://doi.org/10.1007/978-3-642-14623-7_11

Ostrovsky R, Sahai A, Waters B, 2007. Attribute-based encryption with non-monotonic access structures. Proc 14th ACM Conf on Computer and Communications Security, p.195-203. https://doi.org/10.1145/1315245.1315270

Parno B, Raykova M, Vaikuntanathan V, 2012. How to delegate and verify in public: verifiable computation from attribute-based encryption. *LNCS*, 7194:422-439. https://doi.org/10.1007/978-3-642-28914-9_24

Pirretti M, Traynor P, McDaniel P, et al., 2006. Secure attribute-based systems. Proc 13th ACM Conf on Computer and Communications Security, p.99-112. https://doi.org/10.1145/1180405.1180419

Ren K, Wang C, Wang Q, 2012. Security challenges for the public cloud. *IEEE Int Comput*, 16(1):69-73. https://doi.org/10.1109/MIC.2012.14

Sahai A, Waters B, 2005. Fuzzy identity-based encryption. *LNCS*, 3494:457-473. https://doi.org/10.1007/11426639_27

Sahai A, Seyalioglu H, Waters B, 2012. Dynamic credentials and ciphertext delegation for attribute-based encryption. *LNCS*, 7417:199-217. https://doi.org/10.1007/978-3-642-32009-5_13

Shamir A, 1979. How to share a secret. *Commun ACM*, 22(11):612-613. https://doi.org/10.1145/359168.359176

Wang C, Wang Q, Ren K, et al., 2009. Ensuring data storage security in cloud computing. 17th Int Workshop on Quality of Service, p.1-9. https://doi.org/10.1109/IWQoS.2009.5201385

Wang GJ, Liu Q, Wu J, et al., 2011. Hierarchical attribute-based encryption and scalable user revocation for sharing data in cloud servers. *Comput Secur*, 30(5):320-331. https://doi.org/10.1016/j.cose.2011.05.006

Wang ZJ, Huang DJ, 2018. Privacy-preserving mobile crowd sensing in ad hoc networks. *Ad Hoc Networks*, 73:14-26. https://doi.org/10.1016/j.adhoc.2018.02.003

Wang ZJ, Huang DJ, Wu HJ, et al., 2014. Towards distributed privacy-preserving mobile access control. IEEE Global Communications Conf, p.582-587. https://doi.org/10.1109/GLOCOM.2014.7036870

Wang ZJ, Huang DJ, Zhu Y, et al., 2015. Efficient attribute-based comparable data access control. *IEEE Trans Comput*, 64(12):3430-3443. https://doi.org/10.1109/TC.2015.2401033

Waters B, 2011. Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. *LNCS*, 6571:53-70. https://doi.org/10.1007/978-3-642-19379-8_4

Weiss A, 2007. Computing in the clouds. *NetWorker*, 11(4):16-25. https://doi.org/10.1145/1327512.1327513

Xu ZQ, Martin KM, 2012. Dynamic user revocation and key refreshing for attribute-based encryption in cloud storage. 11th IEEE Int Conf on Trust, Security and Privacy in Computing and Communications, p.844-849. https://doi.org/10.1109/TrustCom.2012.136

Xu ZQ, Martin KM, 2013. A practical deployment framework for use of attribute-based encryption in data protection. IEEE 10th Int Conf on High Performance Computing and Communications & IEEE Int Conf on Embedded and Ubiquitous Computing, p.1593-1598. https://doi.org/10.1109/HPCC.and.EUC.2013.224

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, Ren K, Lou WJ, 2008. Attribute-based content distribution with hidden policy. 4th Workshop on Secure Network Protocols, p.39-44. https://doi.org/10.1109/NPSEC.2008.4664879

Yu SC, Wang C, Ren K, et al., 2010. Achieving secure, scalable, and fine-grained data access control in cloud computing. Proc IEEE INFOCOM, p.534-542. https://doi.org/10.1109/INFCOM.2010.5462174