This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2016.2520932, IEEE Transactions on Services Computing

Flexible and Fine-Grained Attribute-Based Data Storage in Cloud Computing

Jiguo Li, Wei Yao, Yichen Zhang, Huiling Qian and Jinguang Han, *Member, IEEE*

Abstract—With the development of cloud computing, outsourcing data to cloud server attracts lots of attentions. To guarantee the security and achieve flexibly fine-grained file access control, attribute based encryption (ABE) was proposed and used in cloud storage system. However, user revocation is the primary issue in ABE schemes. In this article, we provide a ciphertext-policy attribute based encryption (CP-ABE) scheme with efficient user revocation for cloud storage system. The issue of user revocation can be solved efficiently by introducing the concept of user group. When any user leaves, the group manager will update users' private keys except for those who have been revoked. Additionally, CP-ABE scheme has heavy computation cost, as it grows linearly with the complexity for the access structure. To reduce the computation cost, we outsource high computation load to cloud service providers without leaking file content and secret keys. Notbaly, our scheme can withstand collusion attack performed by revoked users cooperating with existing users. We prove the security of our scheme under the divisible computation Diffie-Hellman (DCDH) assumption. The result of our experiment shows computation cost for local devices is relatively low and can be constant. Our scheme is suitable for resource constrained devices.

Index Terms—cloud computing, attribute-based encryption, outsource decryption, user revocation, collusion attack.

1 Introduction

CLOUD computing is regarded as a prospective computing paradigm in which resource is supplied as service over the Internet. It has met the increasing needs of computing resources and storage resources for some enterprises due to its advantages of economy, scalability, and accessibility. Recently, several cloud storage services such as Microsoft Azure and Google App Engine were built and can supply users with scalable and dynamic storage.

With the increasing of sensitive data outsourced to cloud, cloud storage services are facing many challenges including data security and data access control. To solve those problems, attribute-based encryption (ABE) schemes [1-3] have been applied to cloud storage services. Sahai and Waters [1] first proposed ABE scheme named fuzzy identity-based encryption which is derived from identity-based encryption (IBE) [4]. As a new proposed cryptographic primitive, ABE scheme not only has the advantage of IBE scheme, but also provides the characteristic of "one-to-many" encryption. Presently, ABE mainly includes two categories called ciphertext-policy ABE (CP-ABE) [2] and key-policy ABE (KP-ABE) [3]. In CP-ABE, ciphertexts are associated with access policies and user's private keys are associated with attribute sets. A user can decrypt the ciphertext if his attributes satisfy the access policy embedded in the ciphertext. It is contrary in KP-ABE. CP-ABE is more suitable for the outsourcing data architecture than KP-ABE because the access policy is defined by the data owners. In this article, we present an efficient CP-ABE with user revocation ability.

1.1 Related Work

Although ABE has shown its merits, user revocation and attribute revocation are the primary concerns. The revocation problem is even more difficult peculiarly in CP-ABE schemes, because each attribute is shared by many users. This means that revocation for any attribute or any single user may affect the other users in the system. Recently, some work [5-9] has been proposed to solve this problem in efficient ways. Boldyreva et al. [5] presented an IBE scheme with efficient revocation, which is also suitable for KP-ABE. Nevertheless, it is not clear whether their scheme is suitable for CP-ABE. Yu et al. [6] provided an attribute based data sharing scheme with attribute revocation ability. This scheme was proved to be secure against chosen plaintext attacks (CPA) based on DBDH assumption. However, the length of ciphertext and user's private key are proportional to the number of attributes in the attribute universe. In the key generation, encryption and decryption stages, computation involves all attributes in the attribute universe. Hence, it is expensive in communication and computation cost for users. Tysowski et al. [8] gave an easy method to perform user revocation operation by combining CP-ABE with re-encryption. In their scheme, each user belongs to a group and holds a group secret key issued by the group. However, their scheme does not resist collusion attack performed by revoked users cooperating with existing users. The reason is that each user's group secret key is same in the same group. The attributes of the revoked users can be used by the user in the same group without the specified attrib-

- *J. Li, W. Yao, Y. Zhang and H. Qian are with the College of Computer and Information, Hohai University, Nanjing, China 211100. Email: ljg1688@163, lijiguo@hhu.edu.cn*
- *J. Han is with the Jiangsu Provincial Key Laboratory of E-Business, Nanjing University of Finance and Economics, Nanjing, Jiangsu, China 210003. E-mail: jghan22@gmail.com.*

Manuscript received (insert date of submission if desired). Please note that all acknowledgments should be placed at the end of the paper, before the bibliography.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2016.2520932, IEEE Transactions on Services Computing

utes. Additionally, we point out that there is the same security risk in the schemes [7, 9].

Through applying ABE schemes to cloud storage services, we can both ensure the security of stored data and achieve fine-grained data access control. Unfortunately, ABE scheme requires high computation overhead during performing encryption and decryption operations. This defect becomes more severe for lightweight devices due to their constrained computing resources. To reduce the computation cost for resource-constrained devices, some cryptographic operations with high computational load were outsourced to cloud service providers [10-13]. Combined proxy re-encryption with lazy re-encryption technique, Yu et al. [10] designed a KP-ABE scheme with fine-grained data access control. This scheme requires that the root node in the access tree is an AND gate and one child is a leaf node which is associated with the dummy attribute. The dummy attribute is required to be included in every data document's attribute set and will never be updated. In their scheme, cloud service provider stores all the private key components for user's private key except for the one corresponding to the dummy attribute. However, cloud service provider does not learn the plaintext for any data document. Green et al. [11] provided an efficient CP-ABE scheme with outsourcing decryption. In their scheme, user's private key is blinded through using a random number. Both the private key and the random number are kept secret by the user. The user shares his blinded private key to a proxy to perform outsourced decryption operation. In this paper, we use the similar techniques as [10-11] to extend our scheme with outsourcing ability.

However, there is a major limitation to single-authority ABE as in IBE. Namely, each user authenticates him to the authority, proves that he has a certain attribute set, and then receives secret key associated with each of those attributes. Thus, the authority must be trusted to monitor all the attributes. It is unreasonable in practice and cumbersome for authority. Chase [14] designed a multi-authority ABE scheme with central authority. Their scheme is proved secure in the selective attribute model. Liu et al. [15] proposed a fully secure multi-authority CP-ABE which includes multiple central authorities so that no single authority can decrypt any ciphertext. In order to protect privacy of the user, Han et al. [16] presented a decentralized KP-ABE scheme with privacy-preserving. Similarly, Qian et al. [17] provided a decentralized CP-ABE with fully hidden access structure. Furthermore, they [18] proposed a privacy-preserving personal health record using multi-authority ABE with revocation. Recently, some traceable CP-ABE schemes [19-21] were proposed in order to find out an efficient solution to identify malicious users who purposely share their decryption keys.

1.2 Our Motivation and Contribution

Security issues are main obstacles for wide aplication of cloud computing. Recently, Yu et al. [22] presented a multi-keyword top-k retrieval searchable encryption scheme so as to solve data privacy issues. To ensure security for data outsourcing, Yang et al. [23] proposed a secure overlay cloud storage system with ability for file assured deletion and policy-based access control. In this article, we focus on designing a CP-ABE scheme with efficient user revocation for cloud storage system. We aim to model collusion attack performed by revoked users cooperating with existing users. In the scheme [8], when a user leaves from a user group, the group manager only revokes his group secret key which implies that the user's private key associated with attributes is still valid. If someone in the group intentionally exposes the group secret key to the revoked user, he can perform decryption operations through his private key. To clarify this attack, a concrete instance is given. Assume that the data is encrypted under the policy "professor AND cryptography" and the group public key GPK. Suppose that there are two users: $user_1$ and $user_2$ whose private keys are associated with the attribute sets {male, professor, cryptography} and {male, student, cryptography} respectively. If both of them are in the group and hold the group secret key, then $user_1$ can decrypt the data but $user_2$ can't. When $user_1$ is revoked from the group, he can't decrypt alone because he does not have the updated group secret key. However, the attributes of $user_1$ are not revoked and $user_2$ has the updated group secret key. So, $user_1$ can collude with $user_2$ to perform the decryption operation. Furthermore, security model and proof were not provided in their scheme [8].

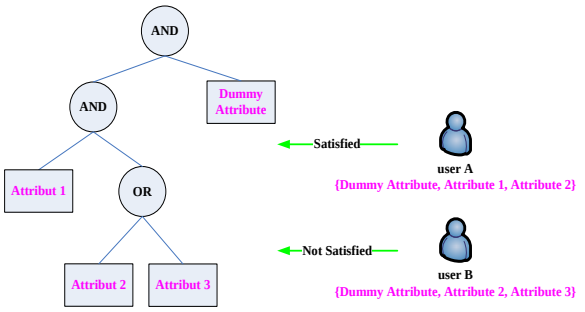

Fig. 1. Access tree used in encryption

Inspired by [6, 8, 24], we give a formal definition and security model for CP-ABE with user revocation. Furthermore, we construct an efficient user revocation CP-ABE scheme through improving the scheme in [8] and prove our scheme is CPA secure under the selective model. To solve above security issue, we embed a certificate into each user's private key. In this way, each user's group secret key is different from others and bound together with his private key associated with attributes. To reduce users' computation burdens, we introduce two cloud service providers named encryption-cloud service provider (E-CSP) and decryption-cloud service provider (D-CSP). The duty of E-CSP is to perform outsourced encryption operation and D-CSP is to perform outsourced decryption operation. As in [10], access tree used in encryption is defined as in Fig. 1. The root node is an AND gate and one child is a leaf node which is associated with the dummy attribute. The dummy attribute is required to be included in every user's attribute set. In Fig. 1, user A's attributes satisfy the access tree while user B's attributes do not. In the encryption phase, the operation associated with the dummy attribute is performed locally while the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2016.2520932, IEEE Transactions on Services Computing

operation associated with the sub-tree is outsourced to E-CSP. To securely outsource decryption operation with heavy bilinear computation, user's private key is blinded as in [11].

1.3 Paper Organization

In section 2, we give preliminaries used throughout this paper. The outline of the scheme is presented in section 3. In section 4, we present a security model of our scheme. In section 5, a concrete construction is given. In section 6, we give a security analysis about our scheme. Section 7 shows an experiment for our scheme. We conclude our paper in section 8.

2 Preliminaries

2.1 Bilinear Pairings [25]

Let G and G_T be cyclic groups with prime order p. A bilinear map $e: G \times G \to G_T$ has the following properties.

(1) Bilinearity. $\forall g_1, g_2 \in G$, $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$, where $a, b \in Z_P^*$.

(2) Non-degeneracy. $e(g_1, g_2) \neq 1$.

(3) Computability. There is an efficient algorithm to compute $e(g_1, g_2)$.

2.2 Divisible Computation Diffie-Hellman (DCDH) Assumption [30]

DCDH Problem. Let G be a group with prime order p. g is a generator in G. For a given tuple $\langle g, g^a, g^b \rangle$ where $a, b \in Z_P^*$. The DCDH problem is to output $g^{a/b}$.

DCDH Assumption. We say that DCDH assumption holds if no probabilistic polynomial time (PPT) adversaries can solve the DCDH problem with at most a negligible advantage.

The DCDH Problem is an equivalent variation of computational Diffie-Hellman problem [30].

2.3 Access Structure [26]

Let $P = \{P_1, P_2, ..., P_n\}$ be a set of parties. A collection $A \subseteq 2^P$ is monotone if $\forall B, C$: if $B \in A$ and $B \subseteq C$ then $C \in A$. An access structure is a collection A for non-empty subsets of P, i.e., $A \in 2^P \setminus \{\varnothing\}$. The sets in A are named the authorized sets, and the sets not in A are named the unauthorized sets.

2.4 Ciphertext-Policy Attribute-Based Encryption

A basic CP-ABE scheme [2] concludes the following fundamental algorithms:

$Setup(\lambda)$: This algorithm takes a security parameter λ as input. It outputs a public parameter PK and a master key MK.

$Encrypt(PK, M, A)$: This algorithm takes the public parameter PK, a message M, and an access policy A in the attribute universe as input. The algorithm outputs a ciphertext CT such that only the user whose attribute set satisfies the access policy can decrypt. We assume that the ciphertext implicitly includes A.

$KenGen(MK, S)$: This algorithm takes the master key MK and an attribute set S as input. It outputs a private key SK with respect to the attribute set S.

$Decrypt(PK, CT, SK)$: This algorithm takes the public parameter PK, a ciphertext CT, and a private key SK as input. If the user's attribute set S satisfies the access structure A embedded in the CT, then the algorithm decrypts the ciphertext successfully and returns M.

2.5 Proxy Re-Encryption

Proxy re-encryption [27] allows an honest-but-curious proxy to convert a ciphertext encrypted by Alice's public key into a new ciphertext that is able to be decrypted by Bob's secret key. In this process, the proxy is unable to get the underlying plaintext. More formally, a proxy re-encryption scheme enables the proxy with the proxy re-encryption key to translate ciphertext encrypted by public key pk_a into ciphertext encrypted by public key pk_b.

3 Outline of CP-ABE with Efficient User Revocation

3.1 Explanation of Symbols

There are seven entities in total in our scheme. They are listed in Table 1.

TABLE 1 ENTITIES

Symbol	Description
TA	Trusted Authority
GM	Trusted Group Manager
DO	Data Owner
DU	Data User
CSS	Cloud Storage Server
E-CSP	Encryption-Cloud Service Provider
D-CSP	Decryption-Cloud Service Provider

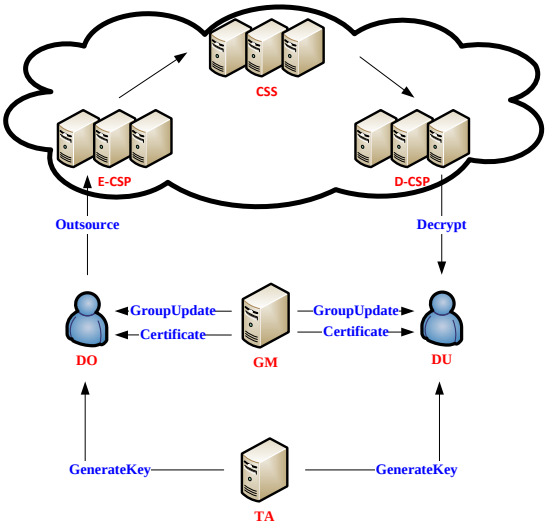

Fig. 2. Our system model

TA is a trusted authority who authenticates users' attribute sets and generates corresponding private keys for them. GM is a trusted group manager who generates certificates for users, updates the private keys of users, and applies CSS for re-encryption operations. CSS in our

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2016.2520932, IEEE Transactions on Services Computing

scheme is a cloud storage server, who is honest-but-curious. To reduce the computation cost for cryptographic operations, we outsource most of encryption operation to E-CSP and decryption operation to D-CSP. Users in the system play two roles: data owner and data user. They are denoted as DO and DU respectively. Our system model is shown in Fig. 2.

3.2 Algorithm Definition

In our CP-ABE scheme with user revocation, we assume that a user's private key includes two parts. One is associated with his authorized attributes and the other one is associated with the group which he belongs to. When one or more users leave the group, GM updates group key pair and updates private keys for existing users. To revoke their access ability to the stored data, GM also applies for re-encryption operations from CSS. A workflow of all algorithms is described in Fig. 3. A CP-ABE scheme with user revocation consists of the following eight algorithms.

$SystemSetup(\lambda) \rightarrow \{MK, PK\}$: This algorithm is performed by TA. It takes a security parameter λ as input and outputs its master key MK and public parameter PK.

$GroupSetup(PK) \rightarrow \{GMK_0, GPK_0, Dic_0\}$: This algorithm is performed by GM. It takes PK as input and outputs the group master key GMK_0, the group public key GPK_0, and a dictionary Dic_0 (initially empty) where 0 denotes initial version. When any user leaves the group, the group key pair and the dictionary will be updated to a new version which increases by 1. In our scheme, we denote current version as ver.

$CertGen(PK, UID, GMK_{ver}) \rightarrow \delta_{ver}$: This algorithm is performed by GM. It takes the public parameter PK, user's identity UID and the group master key GMK_{ver} as input. It outputs a certificate δ_{ver} for user whose identity is UID.

$KeyGen(PK, MK, GPK_{ver}, S, UID, \delta_{ver}) \rightarrow \{DSK_{ver}, UP_{ver}\}$: In this algorithm, TA takes the public parameter PK, the master key MK, the group public key GPK_{ver}, user's attribute set S, user's identity UID and corresponding certificate δ_{ver} as input. It outputs a private key DSK_{ver} and a tuple UP_{ver}. UP_{ver} is used to update private key of the user. TA sends DSK_{ver} to the user. The tuple UP_{ver} is sent to GM and added into the dictionary Dic_{ver}.

$Encrypt(PK, GPK_{ver}, M, \mathrm{A}) \rightarrow CT_{ver}$: This algorithm is performed by DO. It takes the public parameter PK, the group public key GPK_{ver}, a message M, and an access structure A as input. It outputs a ciphertext CT_{ver}.

$GroupUpdate(PK, GMK_{ver}, Dic_{ver}) \rightarrow \{GMK_{ver+1}, GPK_{ver+1}, Re-Key_{ver \rightarrow ver+1}, Dic_{ver+1}\}$: This algorithm is performed by GM. It takes the public parameter PK, the group master key GMK_{ver}, and the dictionary Dic_{ver} as input. It outputs a pair of new group keys $\{GMK_{ver+1}, GPK_{ver+1}\}$, a re-encryption key $Re-Key_{ver \rightarrow ver+1}$ used for re-encryption, and an updated dictionary Dic_{ver+1}. Each tuple in the dictionary Dic_{ver+1} is updated as UP_{ver+1} and sent to corresponding user in the group. After this step, current version for the group is updated as $ver+1$. The current group key pairs $\{GMK_{ver+1}, GPK_{ver+1}\}$ are used in the algorithms $KeyGen()$, and $Encrypt()$.

$UserUpdate(DSK_{ver}, UP_{ver+1}) \rightarrow DSK_{ver+1}$: This algorithm is performed by the existing users in the group. It takes a private key DSK_{ver} and a corresponding tuple UP_{ver+1} as input and outputs an updated private key DSK_{ver+1}.

$Re-Encrypt(CT_{ver}, Re-Key_{ver \rightarrow ver+1}) \rightarrow CT_{ver+1}$: This algorithm is performed by CSS. It takes the ciphertext CT_{ver} and a re-encryption key $Re-Key_{ver \rightarrow ver+1}$ as input and outputs a new ciphertext CT_{ver+1}.

$Decrypt(PK, CT_{ver+1}, DSK_{ver+1}) \rightarrow M$: This algorithm is performed by DU. It takes PK, CT_{ver+1} and DU's private key DSK_{ver+1} as input. If DU is authorized, it outputs the plaintext M. If not, it outputs $\perp$.

4 Security Model

In our security model, the revoked users may collude with the existing users in the same group to attack this group and achieve access to some data. We assume that the revoked users can get private keys that satisfy the specific access structure but the version is not the current version of the attacked group. On the contrary, existing users are able to get private keys that do not satisfy the specific access structure but the version is the current version. To formalize the security model, the following game between adversary A and challenger B is defined.

Init: The adversary A selects the challenge access structure A^*, group identity GID^*, and version number ver^* and sends them to challenger B.

Setup: The challenger B runs $SystemSetup()$ algorithm to get the master key MK and the public parameter PK. B runs $GroupSetup()$ algorithm to get the group maser key GMK_0 and the group public key GPK_0 for GID^*. B then uses $GroupUpdate()$ algorithm to get the group master key GMK_{ver}, the group public key GPK_{ver}, and the re-encryption key $Re-Key_{ver-1 \rightarrow ver}$ from $ver=1$ to ver^*. Finally, B gives the public parameter PK and the group public keys $\{GPK_{ver}\}_{0 \le ver \le ver^*}$ to the adversary A. B keeps the master key MK, the group master keys $\{GMK_{ver}\}_{0 \le ver \le ver^*}$, and the re-encryption keys $\{Re-Key_{ver \rightarrow ver+1}\}_{0 \le ver \le ver^*-1}$.

Phase 1: The adversary A launches many queries as follows. In this process, Type-I query and Type-II query

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2016.2520932, IEEE Transactions on Services Computing

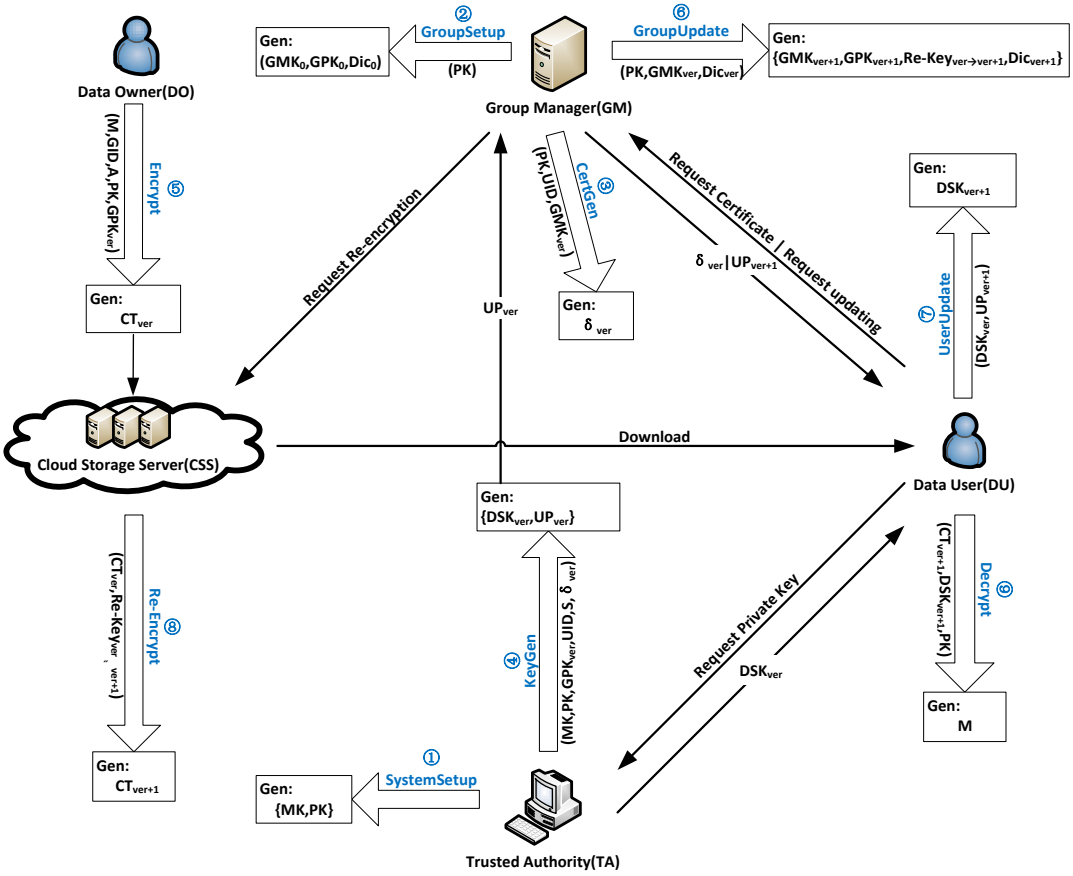

Fig. 3. CP-ABE with efficient user revocation.

respectively formalize the abilities of the revoked users and the existing users.

- Type-I query:

  Certificate query $\langle UID, GID^*, ver \rangle$ where $ver < ver^*$. The challenger B runs the algorithm $CertGen(PK, UID, GMK_{ver})$ to produce a certificate δ_{ver}. B returns δ_{ver} to A.

  Private key query $\langle UID, GID^*, S, \delta_{ver} \rangle$ where the attribute set S satisfies the access policy A^* but the version ver of δ_{ver} must satisfies $ver < ver^*$. The challenger B runs the algorithm $KeyGen(PK, MK, GPK_{ver}, S, UID, \delta_{ver})$ to produce the corresponding private key DSK_{ver}. B returns DSK_{ver} to A.

- Type-II query:

  Certificate query $\langle UID, GID^*, ver^* \rangle$. The challenger B runs the algorithm $CertGen(PK, UID, GMK_{ver^*})$ to produce a certificate δ_{ver^*}. B returns δ_{ver^*} to A.

  Private key query $\langle UID, GID^*, S, \delta_{ver^*} \rangle$ where the attribute set S does not satisfy the access policy A^*. The challenger B runs the algorithm $KeyGen(PK, MK, GPK_{ver^*}, S, UID, \delta_{ver^*})$ to produce the corresponding private key DSK_{ver^*}. B returns DSK_{ver^*} to A.

- User update query $\langle UID, DSK_{ver} \rangle$ where UID has appeared in Type-I private key query and $ver < ver^* - 1$ (If $ver = ver^* - 1$, B will refuse this query because the version number of updated private key will be ver^*). The challenger B responds with a tuple UP_{ver+1} to A. A runs $UserUpdate(DSK_{ver}, UP_{ver+1})$ to get updated private key DSK_{ver+1}.
- Re-encryption query $\langle CT_{ver} \rangle$ where CT_{ver} is the ciphertext and $ver < ver^*$. B responds with ciphertext CT_{ver+1} through running the algorithm $Re-Encrypt(CT_{ver}, Re-Key_{ver \to ver+1})$.

Challenge: Once A checks that Phase 1 is over, it submits two plaintexts M_0 and M_1 with equal length to the challenger B. B selects $\tilde{b} \in \{0,1\}$ at random and computes the challenge ciphertext $CT_{\tilde{b}} = Encrypt(PK, GPK^*, M_{\tilde{b}}, A^*)$. B returns $CT_{\tilde{b}}$ to A.

Phase 2: A launches additional queries as in phase 1 and B answers as in phase 1.

Guess: A outputs a guess $\tilde{b}' \in \{0,1\}$. He wins the game if $\tilde{b} = \tilde{b}'$. The advantage for the adversary A winning the game is defined as $Adv_A = \left| Pr\left[\tilde{b} = \tilde{b}'\right] - \frac{1}{2} \right|$.

Definition 1: We say that a ciphertext-policy attribute-based encryption scheme with user revocation is CPA secure if

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2016.2520932, IEEE Transactions on Services Computing

Adv_A is negligible for any probabilistic polynomial time adversary.

5 Concrete Construction

5.1 Access Tree

Before describing our scheme, we review the concept of access tree proposed in [2-3]. In our construction, private keys are bound to attribute sets and messages are encrypted through access trees. Each interior node in the access tree is a threshold gate and the leave nodes are associated with attributes. A user can decrypt a ciphertext only if his attribute set satisfies the access tree embedded in the ciphertext. We use the same notation as [2-3] to describe the access trees.

Access Tree T **.** Let T be a tree expressing an access policy. Each non-leaf node in the tree expresses a threshold gate, depicted by its children. num_x denotes the number of children in node x and k_x denotes its threshold value, where $0 < k_x \le num_x$. The threshold gate means an OR gate when $k_x = 1$. It means an AND gate when $k_x = num_x$. Each leaf node x in the tree is depicted by an attribute and a threshold value $k_x = 1$.

In the access trees, a few functions are defined. We represent the parent for the node x of the tree by $parent(x)$. In [2-3], the function $att(x)$ is used to represent the attribute associated with the leaf node x of the tree. An order among the children for every node in the access tree T is defined, namely, the children for a node are enumerated from 1 to num_x. The function $index(x)$ denotes a number associated with the node x, where the index values are uniquely assigned to nodes of the access policy for a given ciphertext in an arbitrary manner.

Satisfying an Access Tree. Assume T is an access tree with root R. T_x represents the subtree for T rooted in the node x. Therefore T is represented as T_R. If attribute set S satisfies the access tree T_x, it is denoted as $T_x(S) = 1$. $T_x(S)$ is evaluated recursively as follows: If x is a non-leaf node, we compute $T_{x'}(S)$ for all children x' in node x. $T_x(S)$ outputs 1 if and only if at least k_x children return 1. If x is a leaf node, then $T_x(S)$ outputs 1 if and only if $att(x) \in S$.

5.2 CP-ABE with Efficient User Revocation

Let G and G_T be multiplicative cyclic bilinear groups with large prime order p, and g be a generator of G. Let $e: G \times G \to G_T$ denote the bilinear map. $\Delta_{i,S}(x) = \prod_{j \in S, j \ne i} \frac{x-j}{i-j}$ for $i \in Z_P^*$ is defined as the Lagrange coefficient and the elements of S are in Z_P^*. A hash function $H: \{0,1\}^* \to G$ maps an attribute or an identity to a random group element.

$SystemSetup(\lambda) \to \{MK, PK\}$: TA randomly chooses two exponents $\alpha, \beta \in Z_P^*$ and calculates g^α to produce a master key $MK = \{\beta, g^\alpha\}$. TA then computes g^β, $g^{1/\beta}$, and $e(g,g)^\alpha$. $PK = \{G, g, g^\beta, e(g,g)^\alpha\}$ is published as the public parameter and $g^{1/\beta}$ is sent to GM to perform revocation operations.

$GroupSetup(PK) \to \{GMK_0, GPK_0, Dic_0\}$: GM chooses a random exponent $\gamma_0 \in Z_P^*$ as a group master key $GMK = \{\gamma_0\}$. GM then computes g^{γ_0} and $e(g,g)^{\gamma_0}$. $GPK = \{GID, g^{\gamma_0}, e(g,g)^{\gamma_0}\}$ is published as the group public key, where GID is group identity. Dic_0 is a dictionary which is initially empty. In this algorithm, 0 denotes initial version. When any user leaves the group, the group key pair is updated to a new version. In our scheme, we denote current version as *ver* which is used in later algorithms.

$CertGen(PK, UID, GMK_{ver}) \to \delta_{ver}$: When a user whose identity is UID wants to join a user group, he/ she should request a certificate from the group manager. If GM accepts his/ her request, GM generates a certificate as $\delta_{ver} = g^{\beta\gamma_{ver}} \times H(UID)^{\gamma_{ver}}$. Then, GM sends δ_{ver} to the user through a secure channel.

$KeyGen(PK, MK, GPK_{ver}, S, UID, \delta_{ver}) \to \{DSK_{ver}, UP_{ver}\}$: TA authenticates the user's attributes and executes a private key generation algorithm. This algorithm is divided into two steps:

(1) TA verifies $e(g, \delta_{ver}) = e(g^\beta \times H(UID), g^{\gamma_{ver}})$ to check whether the user is a member in the group or not. If the result of verification algorithm is true then go to the second step. Else, the algorithm outputs $\perp$.

(2) TA performs the following operations to obtain a private key corresponding to the attribute set S and the group identity GID. We assume that the dummy attribute is implicitly included in S.

TA chooses $r_j \in Z_P^*$ for each attribute $j \in S$, $r_g \in Z_P^*$ for the group identity GID and random numbers $r, t_1, t_2 \in Z_P^*$. TA computes $H(UID)^{\gamma_{ver}} = \delta_{ver} / (g^{\gamma_{ver}})^\beta$. Then, he computes the private key as follows:

$$DSK_S = \{D_1 = g^{t_1(\alpha+r)/\beta};\ \forall j \in S:$$
$$D_j = H(UID)^{\gamma_{ver} t_1} \times H(j)^{t_1 r_j}, D_j' = g^{t_1 r_j}\}$$
$$DSK_{GID} = \{D_2 = (g \times H(UID))^{\gamma_{ver} t_2/\beta};$$
$$D_g = g^{t_2 r} \times H(GID)^{t_2 r_g}, D_g' = g^{t_2 r_g}\}$$

TA sends $DSK_{ver} = \{t_1, t_2, DSK_S, DSK_{GID}\}$ to the user by a secure channel. TA sends $UP_{ver} = \{UID, d_1 = H(UID), d_2 = (g \times H(UID))^{1/\beta}\}$ to GM that the user belongs to and GM adds it into the dictionary Dic_{ver}.

$Encrypt(PK, GPK_{ver}, M, \mathrm{T}) \to CT_{ver}$: When DO intends to upload his files to CSS and share them with the users of a specified group, he first defines an access tree T and gets the group public key. Only the users in the group whose attributes satisfy T can perform decryption operation. The access tree is defined as $\mathrm{T} = \mathrm{T}_1 \ AND \ \mathrm{T}_2$ where T_1 is defined by DO according to access policy

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2016.2520932, IEEE Transactions on Services Computing

and T_2 is a leaf node associated with the dummy attribute. The encryption operation contains two steps. The first step is performed by local devices, and the second step is outsourced to E-CSP.

Local encryption: DO chooses $s_1, s_2 \in Z_P^*$ and computes $e(g,g)^{\alpha s_1}$, $e(g,g)^{\gamma_{ver} s_2}$, $g^{\beta s_1}$, $g^{\beta s_2}$, g^{s_1}, and $H(GID)^{s_1}$. For the root node R of T, DO chooses an 1-degree polynomial with $q_R(0) = s_2$, $q_R(1) = s_{2\_1}$, and $q_R(2) = s_{2\_2}$. T_2 only includes one node, which is associated with the dummy attribute. We denote this node as R_2 and associate it with the value $q_R(2)$ equals s_{2_2}. DO generates the local ciphertext:

$$CT_{local} = \{version = ver, T_2, GID, C_{ver} = M \cdot e(g,g)^{\alpha s_1 + \gamma_{ver} s_2},$$
$$C_1 = g^{\beta s_1}, C_2 = g^{\beta s_2}, C_g = g^{s_1}, C_g^{'} = H(GID)^{s_1},$$
$$R_2 : C_y = g^{s_{2\_2}}, C_y^{'} = H(att(y))^{s_{2\_2}}\}$$

Finally, DO sends $\{CT_{local}, T_1, s_{2\_1}\}$ to E-CSP.

Outsourced encryption: When E-CSP receives $\{CT_{local}, T_1, s_{2\_1}\}$ from DO, it performs the following two steps:

(1) $\forall x \in T_1$ (x denotes the node of the tree T_1), E-CSP chooses a random polynomial q_x with degree $d_x = k_x - 1$, where k_x is the secret sharing threshold value in the node x. E-CSP chooses a d_{R_1} degree polynomial with $q_{R_1}(0) = s_{2\_1}$ for the root node R_1 of T_1. $\forall x \in T_1 \setminus R_1$, E-CSP chooses a d_x degree polynomial with $q_x(0) = q_{parent(x)}(index(x))$.

(2) E-CSP generates the outsourced ciphertext: $CT_{out} = \{T_1, \forall y \in Y_1 : C_y = g^{q_y(0)}, C_y^{'} = H(att(y))^{q_y(0)}\}$ where Y_1 is the set of leaf nodes of T_1. Finally, E-CSP combines CT_{local} and CT_{out} to get a complete ciphertext CT_{ver} and stores it in CSS. The complete ciphertext is as follows:

$$CT_{ver} = \{version = ver, T, GID, C_{ver}, C_1, C_2,$$
$$C_g, C_g^{'}, \forall y \in Y : C_y, C_y^{'}\}$$

where $Y = Y_1 \cup \{R_2\}$ is the set for leaf nodes of T.

$GroupUpdate(PK, GMK_{ver}, Dic_{ver}) \rightarrow \{GMK_{ver+1}, GPK_{ver+1}, Re-Key_{ver \rightarrow ver+1}, Dic_{ver+1}\}$: Whenever any user leaves the group, the group key pair is updated to a new version. This algorithm can be finished in two steps:

(1) GM chooses $\gamma_{ver+1} \in Z_P^*$ as a updated group master key $GMK_{ver+1} = \{\gamma_{ver+1}\}$. GM then computes $g^{\gamma_{ver+1}}$ and $e(g,g)^{\gamma_{ver+1}}$. The updated group public key is $GPK_{ver+1} = \{GID, g^{\gamma_{ver+1}}, e(g,g)^{\gamma_{ver+1}}\}$. GM computes a re-encryption key $Re-Key_{ver \rightarrow ver+1} = g^{(\gamma_{ver+1} - \gamma_{ver})/\beta}$ to perform following re-encryption operation.

(2) For each user in the group, GM updates the current tuple UP_{ver} as $UP_{ver+1} = \{UID, d_1 = H(UID)^{\gamma_{ver+1} - \gamma_{ver}}, d_2 = (g \times H(UID))^{(\gamma_{ver+1} - \gamma_{ver})/\beta}\}$ and sends it to the user to perform following user update operation.

$UserUpdate(DSK_{ver}, UP_{ver+1}) \rightarrow DSK_{ver+1}$: On receiving UP_{ver+1}, each user in the group updates his private key by himself. We denote this process as follows:

$$\forall j \in S : D_j = D_j \times d_1^{t_1} = H(UID)^{t_1 \gamma_{ver+1}} \times H(j)^{t_1 r_j}$$
$$D_2 = D_2 \times d_2^{t_2} = (g \times H(UID))^{\gamma_{ver+1} t_2 / \beta}$$

The updated private key is as follows:

$$DSK_S = \{D_1 = g^{t_1(\alpha + r)/\beta}; \forall j \in S :$$
$$D_j = H(UID)^{t_1 \gamma_{ver+1}} \times H(j)^{t_1 r_j}, D_j^{'} = g^{t_1 r_j}\}$$
$$DSK_{GID} = \{D_2 = (g \times H(UID))^{\gamma_{ver+1} t_2 / \beta};$$
$$D_g = g^{t_2 r} \times H(GID)^{t_2 r_g}, D_g^{'} = g^{t_2 r_g}\}$$

$Re-Encrypt(CT_{ver}, Re-Key_{ver \rightarrow ver+1}) \rightarrow CT_{ver+1}$: After updating the keys, GM applies for re-encryption operation from CSS. Therefore, the ciphertexts stored in CSS are no longer decrypted using the keys of prior version. During this process, CSS is unable to decrypt the ciphertexts through re-key. The re-encryption operation is performed by CSS as follows:

$$C_{ver+1} = C_{ver} \cdot e(Re-Key_{ver \rightarrow ver+1}, C_2)$$
$$= M \cdot e(g,g)^{\alpha s_1 + \gamma_{ver} s_2} \cdot e(g^{(\gamma_{ver+1} - \gamma_{ver})/\beta}, g^{\beta s_2})$$
$$= M \cdot e(g,g)^{\alpha s_1 + \gamma_{ver+1} s_2}$$

The new ciphertext is:

$$CT_{ver+1} = \{version = ver + 1, T, GID, C_{ver+1}, C_1, C_2,$$
$$C_g, C_g^{'}, \forall y \in Y : C_y, C_y^{'}\}$$

$Decrypt(CT_{ver+1}, DSK_{ver+1}, PK) \rightarrow M$: DU downloads and decrypts the ciphertext stored in CSS by his virtue of holding the required attributes and belonging to the specified group. During decrypting process, there are a lot of bilinear pairing operations, which are computationally expensive. To reduce the computation cost, we outsource the pairing operations to D-CSP, on the condition that the data content is still protected from being exposed. The decryption operation contains two steps. The first step is that D-CSP performs partial decryption. The second step is that DU decrypts mediate results to get plaintext.

Outsourced decryption: DU sends the ciphertext CT_{ver+1} and his partial private key $\{DSK_S, DSK_{GID}\}$ to D-CSP. D-CSP checks whether DU's attributes satisfy the access tree T and DU is a member in the group defined in the ciphertext. If so, D-CSP performs following three steps.

(1) D-CSP performs a recursive decryption function $DecryptNode(CT_{ver+1}, DSK_S, x)$, where DSK_S is private key associated with a set of attributes S and x is a node in T. Let $i = att(x)$ if the node x is a leaf node. If $i \in S$, the function $DecryptNode(CT_{ver+1}, DSK_S, x)$ is defined as follows:

$$DecryptNode(CT_{ver+1}, DSK_S, x)$$
$$= \frac{e(D_i, C_x)}{e(C_x^{'}, D_i^{'})}$$
$$= \frac{e(H(UID)^{t_1 \gamma_{ver+1}} \times H(i)^{t_1 r_i}, g^{q_x(0)})}{e(H(att(x))^{q_x(0)}, g^{t_1 r_i})}$$
$$= \frac{e(H(UID), g)^{t_1 \gamma_{ver+1} q_x(0)} \cdot e(H(i), g)^{t_1 r_i q_x(0)}}{e(H(i), g)^{t_1 r_i q_x(0)}}$$
$$= e(H(UID), g)^{t_1 \gamma_{ver+1} q_x(0)}$$
$$= F_x$$

If $i \notin S$, $DecryptNode(CT_{ver+1}, DSK_S, x) = \perp$ is defined. The

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2016.2520932, IEEE Transactions on Services Computing

function $DecryptNode(CT_{ver+1}, DSK_S, x)$ performs recursive operation if x is a non-leaf node. For those child nodes z in x, it calls $DecryptNode(CT_{ver+1}, DSK_S, z)$ and denotes the output as F_z. Let S_x represent an arbitrary k_x sized set of child nodes z such that $F_z \neq \perp$. If such set does not exist, it means that the node is not satisfied and the function responds $\perp$. Otherwise, D-CSP computes

$$\begin{aligned} F_x &= \prod_{z \in S_x} F_z^{\Delta_{i,S_x'}(0)} \\ &= \prod_{z \in S_x} \left(e(H(UID), g)^{t_1 \gamma_{ver+1} q_z(0)}\right)^{\Delta_{i,S_x'}(0)} \\ &= \prod_{z \in S_x} \left(e(H(UID), g)^{t_1 \gamma_{ver+1} q_{parent(z)}(index(z))}\right)^{\Delta_{i,S_x'}(0)} \\ &= \prod_{z \in S_x} e(H(UID), g)^{t_1 \gamma_{ver+1} q_x(i) \cdot \Delta_{i,S_x'}(0)} \\ &= e(H(UID), g)^{t_1 \gamma_{ver+1} q_x(0)} \end{aligned}$$

where $i = index(z), S_x' = \{index(z) : z \in S_x\}$.

If DU's attributes satisfy the access tree T, the function $DecryptNode(CT_{ver+1}, DSK_S, x)$ finally obtains the following result:

$$\begin{aligned} F_R &= DecryptNode(CT_{ver+1}, DSK_S, R) \\ &= e(H(UID), g)^{t_1 \gamma_{ver+1} q_R(0)} \\ &= e(H(UID), g)^{t_1 \gamma_{ver+1} s_2} \end{aligned}$$

where R denotes the root node in T.

(2) If DU is a member in the group that defined by DO, D-CSP generates a true result through calling the function $DecryptNode(CT_{ver+1}, DSK_{GID}, GID)$ as follows:

$$\begin{aligned} F_g &= DecryptNode(CT_{ver+1}, DSK_{GID}, GID) \\ &= \frac{e(D_g, C_g)}{e(C_g', D_g')} \\ &= \frac{e(g^{t_2 r} \times H(GID)^{t_2 r_g}, g^{s_1})}{e(H(GID)^{s_1}, g^{t_2 r_g})} \\ &= \frac{e(g,g)^{t_2 r s_1} \cdot e(H(GID), g)^{t_2 r_g s_1}}{e(H(GID), g)^{t_2 r_g s_1}} \\ &= e(g,g)^{t_2 r s_1} \end{aligned}$$

(3) Then D-CSP computes:

$$B_1 = e(C_1, D_1) = e(g^{\beta s_1}, g^{t_1(\alpha+r)/\beta}) = e(g,g)^{\alpha s_1 t_1 + r s_1 t_1}$$

$$\begin{aligned} B_2 &= e(C_2, D_2) = e(g^{\beta s_2}, (g \times H(UID))^{\gamma_{ver+1} t_2 / \beta}) \\ &= e(g,g)^{\gamma_{ver+1} s_2 t_2} \cdot e(g, H(UID))^{\gamma_{ver+1} s_2 t_2} \end{aligned}$$

At last, D-CSP sends $\{C_{ver+1}, F_R, F_g, B_1, B_2\}$ to DU.

Local decryption: On receiving $\{C_{ver+1}, F_R, F_g, B_1, B_2\}$, DU computes $F_R' = F_R^{1/t_1} = e(H(UID), g)^{\gamma_{ver+1} s_2}$, $F_g' = F_g^{1/t_2} = e(g,g)^{r s_1}$, $B_1' = B_1^{1/t_1} = e(g,g)^{\alpha s_1 + r s_1}$, $B_2' = B_2^{1/t_2} = e(g,g)^{\gamma_{ver+1} s_2} \cdot e(g, H(UID))^{\gamma_{ver+1} s_2}$. Then DU recovers the original message through following operation:

$$\begin{aligned} &\frac{C_{ver+1} \cdot F_R' \cdot F_g'}{B_1' \cdot B_2'} \\ &= \frac{M \cdot e(g,g)^{\alpha s_1 + \gamma_{ver+1} s_2} \cdot e(H(UID), g)^{\gamma_{ver+1} s_2} \cdot e(g,g)^{r s_1}}{e(g,g)^{\alpha s_1 + r s_1} \cdot e(g,g)^{\gamma_{ver+1} s_2} \cdot e(g, H(UID))^{\gamma_{ver+1} s_2}} \\ &= \frac{M \cdot e(g,g)^{\alpha s_1 + \gamma_{ver+1} s_2}}{e(g,g)^{\alpha s_1 + \gamma_{ver+1} s_2}} \\ &= M \end{aligned}$$

6 Security Analysis

The main issue in our scheme is to withstand the collusion attack between the revoked users and existing users. Nonetheless, our scheme can resist such attack through embedding user's certificate from GM into the private key for each user.

Theorem 1. Suppose the hash function H is a random oracle. Our CP-ABE with user revocation is secure against chosen plaintext attack under the selective model if the DCDH assumption holds. Concretely, if there exists an adversary A that can break our scheme with the advantage ε after q_{I} Type-I queries and q_{II} Type-II queries, we can construct an algorithm B that can solve the DCDH problem with the advantage:

$$Adv_{\mathrm{A}} \leq \varepsilon \Big/ 2^{(q_{\mathrm{I}} + q_{\mathrm{II}})} \cdot q_{\mathrm{I}} \cdot q_{\mathrm{II}}$$

Proof. The challenger generates the parameter $\{q, G, G_T, e, g\}$ where G, G_T are cyclic groups with large prime order q and g is a generator in G. The challenger randomly selects $a, b \in \mathrm{Z}_P^*$ and submits $(A,B) = (g^a, g^b)$ to B. The goal of algorithm B is to output g^{ab} through interacting with A as follows.

Init: The adversary A selects the challenge access tree T^*, group identity GID^*, and version number ver^* and sends them to algorithm B.

Setup: To produce the public parameter and group public key, algorithm B performs the following steps:

(1) B picks $\alpha, \beta \in \mathrm{Z}_P^*$ randomly and computes g^β, $e(g,g)^\alpha$ to form the public parameter $PK = \{G, g, g^\beta, e(g,g)^\alpha\}$. The corresponding master key is $MK = \{\beta, g^\alpha\}$.

(2) B picks $\gamma_0, \gamma_1, ..., \gamma_{ver^*-1} \in \mathrm{Z}_P^*$ randomly and computes $\{g^{\gamma_{ver}}\}_{0 \leq ver < ver^*} \cup \{B\}_{ver^*}$, $\{e(g,g)^{\gamma_{ver}}\}_{0 \leq ver < ver^*} \cup \{e(g,B)\}_{ver^*}$ to generate the group public key as $GPK_{ver} = \{GID^*, g^{\gamma_{ver}}, e(g,g)^{\gamma_{ver}}\}_{0 \leq ver < ver^*} \cup \{GID^*, B, e(g,B)\}_{ver^*}$. The corresponding group master key is $GMK = \{\gamma_{ver}\}_{0 \leq ver < ver^*} \cup \{b\}_{ver^*}$, where b is unknown to B.

Algorithm B sends the public parameter PK and the group public keys $GPK_{0 \leq ver \leq ver^*}$ to A.

H-queries: A issues queries to the hash function H for user identity, group identity, and attribute. To answer these queries, B keeps three lists H_{UID}^{list}, H_{GID}^{list}, and H_{Att}^{list} which are initially empty.

(1) H_{UID_i} is a query for user identity UID_i. If UID_i already appears in H_{UID}^{list} as $\langle UID_i, H_i, l_i, c_i \rangle$ then B returns $H(UID_i) = H_i$. Otherwise, B flips a fair coin $c_i \in \{0,1\}$ so that $Pr[c_i = 1] = \delta$. If $c_i = 0$, B randomly

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2016.2520932, IEEE Transactions on Services Computing

selects $l_i \in Z_P^*$ and adds the tuple $\langle UID_i, H_i = g^{l_i}, l_i, c_i \rangle$ into H_{UID}^{list}; if $c_i = 1$, B randomly selects $l_i \in Z_P^*$ and adds the tuple $\langle UID_i, H_i = A^{l_i}, l_i, c_i \rangle$ into H_{UID}^{list}. Then, B responds to A with $H(UID_i) = H_i$.

(2) H_{GID_j} is a query for group identity GID_j. If GID_j already appears in H_{GID}^{list} as $\langle GID_j, H_j, m_j \rangle$ then B returns $H(GID_j) = H_j$. Otherwise, B randomly selects $m_j \in Z_P^*$ and adds the tuple $\langle GID_j, H_j = g^{m_j}, m_j \rangle$ into H_{GID}^{list} and responds to A with $H(GID_j) = H_j$.

(3) H_{Att_k} is a query for attribute Att_k. If Att_k already appears in H_{Att}^{list} as $\langle Att_k, H_k, n_k \rangle$ then B returns $H(Att_k) = H_k$. Otherwise, B randomly selects $n_k \in Z_P^*$ and adds the tuple $\langle Att_k, H_k = g^{n_k}, n_k \rangle$ into H_{Att}^{list} and responds to A with $H(Att_k) = H_k$.

Phase 1: In this process, B maintains two lists Key_{I}^{list} and Key_{II}^{list} (initially empty) and responds queries as follows:

- Type-I query:

  Certificate query $\langle UID_i, GID^*, ver \rangle$ where $ver < ver^*$. We assume that UID_i appears in H_{UID}^{list} as $\langle UID_i, H_i, l_i, c_i \rangle$ (If not, B excutes a query himself as in H_{UID_i}). B returns $\delta_{ver} = g^{\beta \cdot \gamma_{ver}} \times A^{l_i \cdot \gamma_{ver}}$ if $c_i = 1$ or $\perp$ if $c_i = 0$.

  Private key query $\langle UID_i, GID^*, S, ver \rangle$ where a set of attributes S satisfies the access tree T^* and $ver < ver^*$. If UID_i already appears in Key_{II}^{list} as $\langle UID_i, DSK_{\mathrm{II}-i} \rangle$ then the algorithm B reports failure and terminates. Otherwise, B gets $\langle UID_i, H_i, l_i, c_i \rangle$ from H_{UID}^{list} and $\langle GID^*, H^*, m^* \rangle$ from H_{GID}^{list}.

  B chooses $r_j \in Z_P^*$ for each attribute $j \in S$, $r_g \in Z_P^*$ for the group identity GID^* and random number $r, t_1, t_2 \in Z_P^*$. B computes the private key as follows:

$$DSK_S = \{D_1 = g^{t_1 \cdot (\alpha + r)/\beta}; \forall j \in S : D_j = A^{l_i \cdot \gamma_{ver} \cdot t_1} \times H(j)^{r_j \cdot t_1}, D_j' = g^{r_j \cdot t_1}\}$$

$$DSK_{GID^*} = \{D_2 = (g \times A^{l_i})^{t_2 \cdot \gamma_{ver}/\beta}; D_g = g^{r \cdot t_2} \times (H^*)^{r_g \cdot t_2}, D_g' = g^{r_g \cdot t_2}\}$$

  B sends $DSK_{\mathrm{I},i} = \{t_1, t_2, DSK_S, DSK_{GID^*}\}$ to the adversary A and adds $\langle UID_i, DSK_{\mathrm{I},i} \rangle$ into Key_{I}^{list}.

- Type-II query:

  Certificate query $\langle UID_i, GID^*, ver^* \rangle$. We assume that UID_i appears in H_{UID}^{list} as $\langle UID_i, H_i, l_i, c_i \rangle$ (If not, B excutes a query himself as in H_{UID_i}). Otherwise, B returns $\delta_{ver^*} = B^{\beta} \times B^{l_i}$ if $c_i = 0$ or $\perp$ if $c_i = 1$.

  Private key query $\langle UID_i, GID^*, S, ver \rangle$ where a set of attributes S does not satisfy the access tree T^* and $ver = ver^*$. If UID_i already appears in Key_{I}^{list} as $\langle UID_i, DSK_{\mathrm{I},i} \rangle$, B outputs failure and terminates. If UID_i already appears in Key_{II}^{list} as $\langle UID_i, DSK_{\mathrm{II},i} \rangle$, B replies A with $DSK_{\mathrm{II},i}$. Otherwise B gets $\langle UID_i, H_i, l_i, c_i \rangle$ from H_{UID}^{list} and $\langle GID^*, H^*, m^* \rangle$ from H_{GID}^{list}. B chooses $r_j \in Z_P^*$ for each attribute $j \in S$, $r_g \in Z_P^*$ for the group identity GID^* and random number $r, t_1, t_2 \in Z_P^*$. B calculates the private key as follows:

$$DSK_S = \{D_1 = g^{t_1 \cdot (\alpha + r)/\beta}; \forall j \in S : D_j = B^{l_i \cdot t_1} \times H(j)^{r_j \cdot t_1}, D_j' = g^{r_j \cdot t_1}\}$$

$$DSK_{GID^*} = \{D_2 = B^{t_2/\beta} \times B^{t_2 \cdot l_i/\beta}; D_g = g^{r \cdot t_2} \times (H^*)^{r_g \cdot t_2}, D_g' = g^{r_g \cdot t_2}\}$$

  B sends $DSK_{\mathrm{II},i} = \{t_1, t_2, DSK_S, DSK_{GID^*}\}$ to the adversary A and adds $\langle UID_i, DSK_{\mathrm{II},i} \rangle$ into Key_{II}^{list}.

- User update query $\langle UID_i, ver, DSK_{\mathrm{I},i} \rangle$ where UID_i has appeared in Type-I private key query and $ver < ver^* - 1$. B gets $\langle UID_i, H_i, l_i, c_i \rangle$ from H_{UID}^{list} and computes $d_1 = A^{l_i \cdot (\gamma_{ver+1} - \gamma_{ver})}$, $d_2 = (g \times A^{l_i})^{(\gamma_{ver+1} - \gamma_{ver})/\beta}$. Then, B sends $UP_{ver+1} = \langle UID_i, d_1, d_2 \rangle$ to the adversary A. A updates his private key through running the algorithm $UserUpdate(DSK_{\mathrm{I},i}, UP_{ver+1})$.

- Re-encryption query $\langle CT_{ver}, GID^* \rangle$ where CT_{ver} is the ciphertext and $ver < ver^*$. B first computes the re-encryption key as $Re-Key_{ver \to ver+1} = g^{(\gamma_{ver+1} - \gamma_{ver})/\beta}$ and runs $Re-Encrypt(CT_{ver}, Re-Key_{ver \to ver+1})$ to generate a new ciphertext CT_{ver+1}. Then B responds with CT_{ver+1} to the adversary A.

Challenge: Once A checks that Phase 1 ends, it submits two plaintexts M_0 and M_1 with equal length to B. The algorithm B randomly selects $\tilde{b} \in \{0,1\}$ and performs as follows:

(1) B randomly chooses $s_1, s_2 \in Z_P^*$ and gets $\langle GID^*, H^*, m^* \rangle$ from H_{GID}^{list}.

(2) B sets the challenge ciphertext as:

$$CT_b = \{version = ver^*, T^*, GID^*, C_b = M_b \cdot e(g,g)^{\alpha s_1} \cdot e(g,B)^{s_2},$$
$$C_1 = g^{\beta s_1}, C_2 = g^{\beta s_2}, C_g = g^{s_1}, C_g' = (H^*)^{s_1},$$
$$\forall y \in Y : C_y = g^{q_y(0)}, C_y' = H(att(y))^{q_y(0)}\}$$

where, Y is the set for leaf nodes of T^*, $q_y(0)$ of each $y \in Y$ is produced as in the encryption algorithm. Finally, B sends CT_b to the adversary A.

Phase 2: A proceeds to launch queries not issued in Phase 1 and B replies as in phase 1.

Guess: At last, B ignores A 's output and selects a $DSK_{\mathrm{I},i}$ randomly which is a response in Type-I query and a $DSK_{\mathrm{II},i}$ randomly which is a response in Type-II

query. The theoretical evidence is that if A is able to decrypt the challenge ciphertext $CT_{\tilde{b}}$, he must have get two keys that D_2 of $DSK_{\mathrm{II},i}$ matches D_j of $DSK_{\mathrm{I},i}$. We denote $\langle UID_i, H_i, l_i, c_i = 1\rangle$ corresponds to $DSK_{\mathrm{I},i}$ and $\langle UID_i', H_i', l_i', c_i' = 0\rangle$ corresponds to $DSK_{\mathrm{II},i}$. We formulate above theory evidence as $B^{l_i'} = (H_i)^{\gamma_{ver}}$. The equation is equivalent to $g^{b \cdot l_i'} = g^{a \cdot l_i \cdot \gamma_{ver}}$. So, B finally outputs $g^{a/b} = g^{l_i'/l_i \cdot \gamma_{ver}}$ as his answer.

If B does not return $\perp$ during phase 1 then A 's view is identical to its view in the real attack. To complete the proof it remains to calculate the probability that B succees during the simulation. Suppose A makes q_{I} Type-I queries and q_{II} Type-II queries. Then the probability that B does not return $\perp$ in phases 1 is $\delta^{q_{\mathrm{I}}} \cdot (1-\delta)^{q_{\mathrm{II}}}$ (to make it brief, we assume that $q_{\mathrm{I}} = q_{\mathrm{II}}$). This value is maximized when $\delta = 1/2$. Therefore, the probability that B succees during the simulation is $1/2^{(q_{\mathrm{I}}+q_{\mathrm{II}})}$. The probability that B picks out the correct $DSK_{\mathrm{I},i}$ and $DSK_{\mathrm{II},i}$ is $1/q_{\mathrm{I}} \cdot q_{\mathrm{II}}$. This shows that B 's advantage is at most $\varepsilon / 2^{(q_{\mathrm{I}}+q_{\mathrm{II}})} \cdot q_{\mathrm{I}} \cdot q_{\mathrm{II}}$.

7 Efficience Analysis and Experiment

7.1 Efficiency Analysis

We implement our scheme and the scheme [8] on Windows system with an Intel Core i3 CPU 2.13GHz and 2.00GB RAM. In this process, java pairing-based cryptography (JPBC) library version 2.0.0 [28] is used. It is a port of the pairing-based cryptography (PBC) library [29] in C. A comparison of component size between our scheme and the scheme [8] is shown in Table 2, where Nu, Na, and Nt denote the number of data owners, user's attributes and leaf nodes in access tree, respectively. In Table 2, our scheme needs a little more storage space than scheme [8], but it is so slight with the increasing of user's attributes and the growing of the access tree complexity. However, the size for public key of the scheme [8] is associated with the number of data owners while the size for public key of our scheme is constant.

TABLE 2 COMPONENT SIZE (BYTES)

	PK	GPK	private key	ciphertext	Re-key
Scheme [8]	763+264 Nu	132	160+288 Na	1128+296(Nt-2)	132
Our Scheme	1023	282	602+288 Na	1286+296(Nt-2)	132

To compare our scheme with the scheme [8] in actual operation, we run all the algorithms of the two schemes five times respectively and calculate average values. We show them in Fig. 4. All the algorithms are coded using java under a default condition that the access tree contains fifty leaf nodes and the private key satisfies the access tree. From Fig. 4, we find out that our scheme is similar with the scheme [8] in efficiency. Additionally, our scheme also needs two exponent operations (one of which can be pre-computed) and one multiplication operation to generate certificate, which cost about 50 milliseconds. Considering that our scheme resists collusion attack performed by the revoked users cooperating with existing users while the scheme [8] does not, our scheme is more practical.

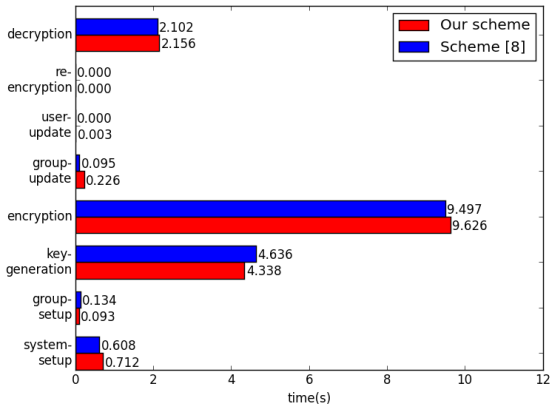

Fig. 4. Running time of all algorithms

Here, we pay more attention on encryption and decryption operations because they are performed by users. We perform encryption and decryption algorithms of our scheme and the scheme [8] on PC. In this process, the leaf node number of access tree is selected ranging from 1 to 100. In Fig. 5 and Fig. 6, the time of our scheme and the scheme [8] linearly grows with the complexity of the access tree. An encryption operation under access tree with 100 attributes takes about 20 seconds and its corresponding decryption operation takes about 4 seconds. In practice, the user's devices probably are computation resource constrained such as mobile phones. The same process will cost more time for these devices.

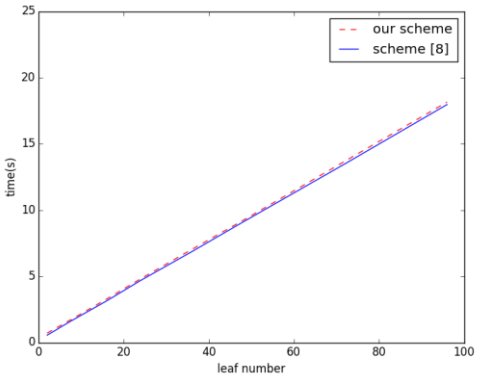

Fig. 5. Encryption time

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2016.2520932, IEEE Transactions on Services Computing

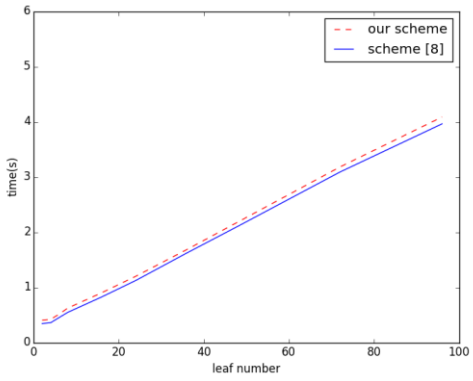

Fig. 6. Decryption time

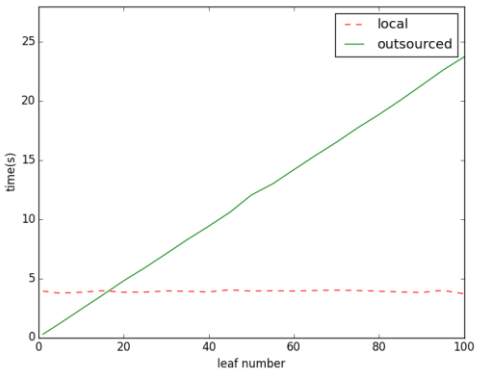

Fig. 7. Encryption time

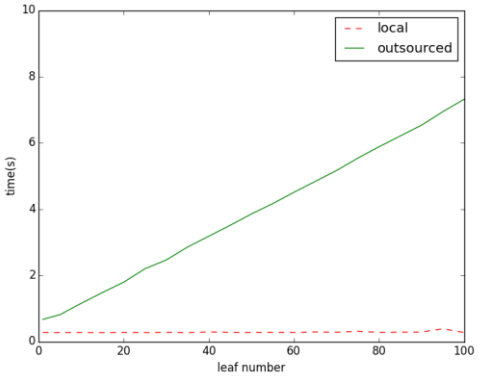

Fig. 8. Decryption time

7.2 Application

To reduce the heavy computation burden on users, we introduce the outsourcing technique into our scheme. We outsource most of computation load to E-CSP and D-CSP and leave very small computation cost to local devices. To prove that our scheme is efficient for resource constrained devices, we transplant our code on an android platform MOTOROLA XT615 with a single-core 800MHz and 512MB RAM. To simulate the factual environment, we develop a simple image storage application which contains a client and a server. The server is deployed on PC to simulate cloud service providers including E-CSP and D-CSP. In fact, actual cloud service providers are much stronger than our PC in computation ability. So, we pay more attention on the computation cost performed on mobile phones.

In our application, we use AES algorithm to encrypt image. Our ABE scheme is used to encapsulate AES key, which achieves fine-grained access control and efficient user revocation. In Fig. 7 and Fig. 8, the time used on server linearly grows with the growing of the access tree complexity, which also indicates that burdensome operation is outsourced to E-CSP and D-CSP. In practice, the time used on server is quite small because actual cloud service providers are much strong in computation ability. The red lines show that local encryption uses about 3.8s and local decryption uses about 300ms, which are relatively fixed and much short. Through such experiment, we prove our scheme is efficient for resource constrained devices such as mobile phones. In conclusion, our scheme can be used in cloud storage system that requires the abilities of user revocation and fine-grained access control.

8 Conclusion

In this article, we provided a formal definition and security model for CP-ABE with user revocation. We also construct a concrete CP-ABE scheme which is CPA secure based on DCDH assumption. To resist collusion attack, we embed a certificate into the user's private key. So that malicious users and the revoked users do not have the ability to generate a valid private key through combining their private keys. Additionally, we outsource operations with high computation cost to E-CSP and D-CSP to reduce the user's computation burdens. Through applying the technique of outsource, computation cost for local devices is much lower and relatively fixed. The results of our experiment show that our scheme is efficient for resource constrained devices.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China (61272542, 61300213), the Fundamental Research Funds for the Central Universities (2013B07014), the Priority Academic Program Development of Jiangsu Higher Education Institutions.

References

[1] A. Sahai and B. Waters, "Fuzzy Identity-Based Encryption," *EUROCRYPT '05*, LNCS, vol. 3494, pp. 457-473, 2005.

[2] J. Bethencourt, A. Sahai and B. Waters, "Ciphertext-Policy Attribute-Based Encryption," *Proc. IEEE Symposium on Security and Privacy*,

pp. 321-334, May 2007, doi: 10.1109/ SP.2007.11.

[3] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data," *Proc. 13th ACM Conference on Computer and Communications Security (CCS '06)*, pp. 89-98, 2006, doi:10.1145/ 1180405.1180418.

[4] D. Boneh and M.K. Franklin, "Identity-Based Encryption from the Weil Pairing," *SIAM Journal of Computing*, vol. 32, no. 3, pp. 586-615, 2003.

[5] A. Boldyreva, V. Goyal, and V. Kumar, "Identity-Based Encryption with Efficient Revocation," *Proc. 15th ACM conference on Computer and communications security (CCS '08)*, pp. 417-426, 2008.

[6] S. Yu, C. Wang, K. Ren, and W. Lou, "Attribute Based Data Sharing with Attribute Revocation," *Proc. 5th ACM Symposium on Information, Computer and Communications Security (ASIACCS '10)*, pp. 261-270, 2010.

[7] M. Yang, F. Liu, J. Han, and Z. Wang, "An Efficient Attribute based Encryption Scheme with Revocation for Outsourced Data Sharing Control," *Proc. 2011 International Conference on Instrumentation, Measurement, Computer, Communication and Control*, pp. 516-520, 2011.

[8] P.K. Tysowski and M.A. Hasan, "Hybrid Attribute-Based Encryption and Re-Encryption for Scalable Mobile Applications in Clouds," *IEEE Transactions on Cloud Computing*, pp. 172-186, 2013.

[9] J. Hur and D. K. Noh, "Attribute-Based Access Control with Efficient Revocation in Data Outsourcing Systems," *IEEE Transactions on Parallel and Distributed Systems*, pp. 1214-1221, 2011.

[10] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving Secure, Scalable, and Fine-Grained Data Access Control in Cloud Computing," *Proc. of IEEE INFOCOM '10*, pp. 1-9, 2010.

[11] M. Green, S. Hohenberger and B. Waters, "Outsourcing the decryption of ABE ciphertexts," *Proc. 20th USENIX Conference on Security (SEC '11)*, pp. 34, 2011.

[12] J. Li, X.F. Chen, J.W. Li, C.F. Jia, J.F. Ma and W.J. Lou, "Fine-Grained Access Control System Based on Outsourced Attribute-Based Encryption," *Proc. 18th European Symposium on Research in Computer Security (ESORICS '13)*, LNCS 8134, Berlin: Springer-Verlag, pp. 592-609, 2013.

[13] J.W. Li, C.F. Jia, J. Li and X.F. Chen, "Outsourcing Encryption of Attribute-Based Encryption with Mapreduce," *Proc. 14th International Conference on Information and Communications Security (ICICS '12)*, LNCS 7618, Berlin: Springer-Verlag, pp. 191-201, 2012. doi: 10.1007/ 978-3-642-34129-8_17

[14] M. Chase, "Multi-authority Attribute Based Encryption," *Proc. 4th Theory of Cryptography Conference (TCC '07)*, LNCS 4392, Berlin: Springer-Verlag, pp. 515-534, 2007.

[15] Z. Liu, Z. Cao, Q. Huang, D. S. Wong and T. H. Yuen, "Fully Secure Multi-Authority Ciphertext-Policy Attribute-Based Encryption without Random Oracles," *Proc. 16th European Symposium on Research in Computer Security (ESORICS '11)*, LNCS 6879, Berlin: Springer-Verlag, pp. 278-297, 2011.

[16] J.G. Han, W. Susilo, Y. Mu and J. Yan, "Privacy-Preserving Decentralized Key-Policy Attribute-Based Encryption," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no.11, pp. 2150-2162, Nov 2012, doi: 10.1109/ TPDS.2012.50.

[17] H.L. Qian, J.G. Li and Y.C. Zhang, "Privacy-Preserving Decentralized Ciphertext-Policy Attribute-Based Encryption with Fully Hidden Access Structure," *Proc. 15th International Conference on Information and Communications Security (ICICS '13)*, LNCS 8233, Berlin: Springer-Verlag, pp. 363-372, 2013.

[18] H.L. Qian, J.G. Li, Y.C. Zhang and J.G. Han, "Privacy Preserving Personal Health Record Using Multi-Authority Attribute-Based Encryption with Revocation," *International Journal of Information Security*, doi: 10.1007/ s10207-014-0270-9.

[19] Z. Liu, Z.F. Cao and Duncan S. Wong, "Black-Box Traceable CP-ABE: How to Catch People Leaking Their Keys by Selling Decryption Devices on eBay," *Proc. 2013 ACM SIGSAC Conference on Computer and Communications Security (CCS '13)*, pp. 475-486, 2013, doi: 10.1145/ 2508859.2516683.

[20] Z. Liu, Z.F. Cao and Duncan S. Wong, "White-Box Traceable Ciphertext-Policy Attribute-Based Encryption Supporting Any Monotone Access Structures," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 1, pp. 76-88, 2013, doi: 10.1109/ TIFS.2012.2223683.

[21] J.T. Ning, Z.F. Cao, X.L. Dong, L.F. Wei and X.D. Lin, "Large Universe Ciphertext-Policy Attribute-Based Encryption with White-Box Traceability," *Proc. 19th European Symposium on Research in Computer Security (ESORICS '14)*, LNCS 8713, Berlin: Springer-Verlag, pp. 55-72, 2014.

[22] J.D. Yu, P. Lu, Y.M. Zhu, G.T. Xue and M.L. Li, "Toward Secure Multikeyword Top-k Retrieval over Encrypted Cloud Data," *IEEE Transactions on Dependable and Secure Computing*, vol. 10, no. 4, pp. 239-250, 2013, doi:10.1109/ TDSC.2013.9

[23] T. Yang, P.P.C. Lee, J.C.S. Lui, R. Perlman, "Secure Overlay Cloud Storage with Access Control and Assured Deletion," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 6, pp. 903-916, 2012, doi: 10.1109/ TDSC.2012.49

[24] L. Cheung and C. Newport, "Provably Secure Ciphertext Policy ABE," *Proc. 14th ACM Conference on Computer and Communications Security (CCS '07)*, pp. 456-465, 2007, doi:10.1145/ 1180405.1180418.

[25] D. Boneh and M.K. Franklin, "Identity-Based Encryption from the Weil Pairing," *CRYPTO '01*, LNCS, vol. 2139, pp. 213-229, Aug. 2001.

[26] A. Beimel, "Secure Schemes for Secret Sharing and Key Distribution" PhD thesis, Israel Institute of Technology, 1996.

[27] M. Blaze, G. Bleumer and M. Strauss, "Divertible Protocols and Atomic Proxy Cryptography," *Proc. International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT '98)*, LNCS 1403, Berlin: Springer-Verlag, pp. 127-144, 1998.

[28] A.D. Caro, "Java Pairing-Based Cryptography Library," http:/ / gas.dia.unisa.it/ projects/ jpbc, 2013.

[29] B. Lynn, "Pairing-Based Cryptography (PBC) Library," http:/ / crypto.stanford.edu/ pbc, 2013.

[30] H.D Robert, F. Bao, H. Zhu, "Variations of Diffie-Hellman Problem," *Proc. 5th International Conference on Information and Communications Security (ICICS '03)*, LNCS 2836, Springer-verlag, pp. 301-312, 2003.

Jiguo Li received his B.S. degree in mathematics from Heilongjiang University, Harbin, China in 1996, M.S. degree in mathematics and Ph.D. degree in computer science from Harbin Institute of Technology, Harbin, China in 2000 and 2003, respectively. During 2006.9-2007.3, he was a visiting scholar at Centre for Computer and Information Security Research, School of Computer Science & Software Engineering, University of Wollongong, Australia. During 2013.2-2014.1, he was a visiting scholar in Institute for Cyber Security in the University of Texas at San Antonio. He is currently a Professor with the College of Computer and Information, Hohai University, Nanjing, China. His research interests include cryptography and information security, cloud computing, wireless security and trusted computing etc. He has published over 100 research papers in refereed international conferences and journals. His work has been cited more than 1200 times at Google Scholar. He has served as program committee member in over 20 international conferences and served as the reviewers in over 50 international journals and conferences.

Wei Yao received his B.S. degree in computer science and technology from China University of Mining and Technology, Xuzhou, China in 2013. He is currently a M.S. student in the College of Computer and Information Engineering, Hohai University, Nanjing, China. His research interests include cryptography and information security, network security, and cloud computing.

Yichen Zhang received her B.S. degree in computer science from the Qiqihar University, Qiqihar, China in 1995. She is currently a lecturer and pursuing the Ph.D. degree in the College of Computer and Information, Hohai University, Nanjing, China. Her research interests include cryptography, network security. She has published over 30 research papers in refereed international conferences and journals.

Huiling Qian received her B.S. degree in communication engineering and M.S. degree in signal and information processing from Hohai University, Nanjing, China in 2011 and 2014, respectively. She currently works as an engineer in Alcatel-Lucent. Her research interests include cryptography and information security, network security, and cloud computing.

Jinguang Han received his Ph.D. degree from University of Wollongong, Australia, in 2013. He currently is an associate professor in Jiangsu Provincial Key Laboratory of E-Business, Nanjing University of Finance and Economics, China. His main research interests include cryptography and information security. He has served as a program committee member in over 10 international conferences. He has published more than 20 research papers in refereed international journals and conferences. He is a member of the IEEE.