Advance Access publication on 2 September 2011
doi:10.1093/comjnl/bxr080

Enforcing Role-Based Access Control for Secure Data Storage in the Cloud

LAN ZHOU*, VIJAY VARADHARAJAN AND MICHAEL HITCHENS

Information and Networked Systems Security Research, Department of Computing, Macquarie University, North Ryde, NSW 2109, Australia
**Corresponding author: vijay.varadharajan@mq.edu.au*

In recent times, there has been increasing interest in storing data securely in the cloud environment. To provide owners of data stored in the cloud with flexible control over access to their data by other users, we propose a role-based encryption (RBE) scheme for secure cloud storage. Our scheme allows the owner of data to store it in an encrypted form in the cloud and to grant access to that data for users with specific roles. The scheme specifies a set of roles to which the users are assigned, with each role having a set of permissions. The data owner can encrypt the data and store it in the cloud in such a way that only users with specific roles can decrypt the data. Anyone else, including the cloud providers themselves, will not be able to decrypt the data. We describe such an RBE scheme using a broadcast encryption algorithm. The paper describes the security analysis of the proposed scheme and gives proofs showing that the proposed scheme is secure against attacks. We also analyse the efficiency and performance of our scheme and show that it has superior characteristics compared with other previously published schemes.

Keywords: role-based access control; encryption; secure data storage; cloud computing

Received 1 February 2011; revised 15 June 2011
Handling editor: Hua Wang

1. INTRODUCTION

Cloud computing has attracted much attention in recent times because of its ability to deliver resources such as computing and storage to users on demand in a cost-effective manner. Due to the continuous growth in the amount of digital information that needs to be stored, there is a clear incentive for the service providers to explore outsourcing of users' data to the cloud. Potentially there could be several benefits to storing data in the cloud. The cloud can provide a scalable high-performance storage architecture, and can help us to significantly reduce the cost of maintenance of individual services.

Since the public cloud[1] is an open platform, and can be subjected to malicious attacks from both insiders and outsiders, the need to protect the privacy and security of the data in the cloud becomes a critical issue. An important aspect when the data are stored in the cloud is: who is able to access and view the data? Consider the scenario where a user (the owner of the data) uses a cloud service provider to store his/her data such as photos in a cloud. Several security requirements can arise in such a scenario. The user (owner of the data) may wish to restrict which other users are able to access and view the data. For instance, a user may want to allow the photo studio (where the user wants to print the photos) and his/her family to access and view the photos stored in the cloud but not anyone else. This places restrictions on the type of users (e.g. outsiders) accessing and viewing the photos. Later, if the user finds another photo studio that prints the photos cheaper, then the user may want to revoke the access permission for the photos from the previous photo studio, and grant the permission to the new one. Moreover the user (owner of the data) may not want the cloud provider to view the photos that the user is storing in the cloud. Here the cloud provider can include the employees of the cloud provider organization (that is, the insiders). The greater the sensitivity of the data stored in the cloud, the more stringent are the security requirements on the outsiders and insiders. For instance, storage of electronic patient records in a healthcare system would warrant such stringent security needs.

One approach is to use cryptographic techniques to encrypt the data before storing it in the cloud. This would allow only the users who have access to the key(s) to decrypt the data and to view the data in the plain form. The problem of achieving secure

[1] In this paper, when we use the word cloud, we are referring to a public cloud.

access to data stored in the cloud is transformed into the problem of access to keys. Some form of access control is then needed to determine who can have access to keys. For instance, in the case of online patient records, a patient can encrypt his/her medical record and then store it in the cloud. The patient can then specify an access control policy that describes who can access and view the record. For instance, this could be his/her doctor, appropriate medical specialist(s) as well as some selected family members. Then we require a secure way of achieving key distribution to these selected users, who can have the access to view the data according to the access control policy. The users who are not allowed by the access control policy cannot view the data as they will not have the key.

Moreover this restriction also applies to the cloud providers themselves. That is, the cloud providers themselves are subjected to these controls, thereby ensuring that they do not have access to stored information in their cloud without the explicit granting of the access privilege by the owner. This situation is different from traditional access control systems, where access control enforcement is carried out by a trusted party, which is usually the service provider. The cloud provider may not be trusted due to the distributed nature of the cloud computing architecture. Hence, users cannot rely on the service provider in the cloud environment to enforce the access control. In fact, the cloud providers themselves may wish to have such a feature to prove (to the owner of the data) that they do not have access to (or the ability to modify) the plain data and hence reduce their liability. Furthermore, there must be mechanisms for allowing new users to be added who can access the data as well as for removing (revoking) users from accessing the data in the cloud.

A trivial solution is that the owner can employ a broadcast encryption (BE) scheme to encrypt the private data and distribute the decryption key to the users, with whom he/she wishes to share the data, via a secure channel. However, in the real world, there could be a large number of owners who may want to store their private data in the cloud as well as a large number of users who may want to access the stored data. If a user needs to obtain a key for each owner who wants him/her to access the private data, the number of keys that each user needs to keep could become very large. Furthermore, when the owner wants to revoke the permission from existing users, a new key needs to be distributed to all the users who are not revoked, to decrypt future messages. This would lead to significant inefficiencies in the implementation of a large-scale system and poor performance.

In this paper, we propose a hybrid scheme that combines access control with cryptography and key distribution to address security requirements for data storage in the cloud. We refer to our hybrid scheme as role-based encryption (RBE) as it involves the use of cryptography with role-based access control to simplify security management. The owner of data encrypts the data in such a way that only the users with appropriate roles as specified by a role-based access control policy can decrypt and view the data. Recall that the central notion of role-based access control is that permissions are associated with roles, and users are assigned to appropriate roles, which simplifies the management of permissions. Roles are created for the various subsets of users based on users' responsibilities and qualifications. A role manager is used to assign a role to a user, or revoke a role from a user. The owner of data can grant permissions to the roles while adding new data, and can also revoke the permissions from the roles on some existing data as needed. Even the cloud provider (who stores the data) will not be able to see the content of the data if the provider is not given the appropriate role. A user is able to join a role after the owner has encrypted the data for that role, and the user will be able to access that data from then on, and the owner does not need to re-encrypt the data. A user can be revoked at any time (e.g if he/she is found to be malicious), in which case the revoked user will not have access to any future data encrypted to this role.

In addition, our RBE scheme is able to deal with role hierarchies, whereby roles can inherit permissions from other roles. In the hierarchical structure of roles, a role can have sub-roles. If a role A inherits all the permissions that role B has, then we say role A is a predecessor role of role B, and role B is a successor role of role A. We will also show how our scheme is able to deal with a user belonging to multiple roles in an effective manner. A significant feature of the proposed scheme is that the decryption key size still remains constant, regardless of the number of roles that the user has been assigned to.

In this paper, we construct such an RBE scheme using a BE mechanism described in [1]. In our scheme, the ciphertext and the decryption key that the user needs to keep is constant in size, and the user can be revoked from the role without affecting the owners and other users of the same role.

The paper is organized as follows. In Section 2, we briefly describe related works that are relevant for our proposed scheme in this paper and consider their limitations. Section 3 gives the definitions of the proposed RBE scheme and describes the security properties of the scheme and the assumptions made. In Section 4, we describe our RBE scheme using ID-based BE mechanism given in [1]. We give an analysis of the security and performance of our scheme in Section 5. In Section 6, we discuss some design aspects that can be optimized to achieve efficient practical implementation of our scheme. Finally, Section 7 concludes the paper.

2. RELATED WORK

In recent years, several schemes have been proposed that use cryptographic techniques to enforce access control such as those used for securing outsourced data on semi-trusted servers given in [2–4]. These schemes transform the access control into a key management problem, and use some form of hierarchical key

management scheme [5]. However, these solutions have several limitations. For instance, if there is a large number of users and owners involved, the overhead involved in setting up the key infrastructure can be very high. In addition, when a user's permission is revoked, all the remaining users in the same role will be affected and their keys need to be changed, which makes these schemes impractical.

Another approach involves the use of the technique called attribute-based encryption (ABE). The first ABE scheme was proposed in [6], in which ciphertexts are labeled with sets of attributes and private keys are associated with access structures that control which ciphertexts a user is able to decrypt. Hence this scheme is also referred to as key-policy ABE or KP-ABE. In KP-ABE scheme, the owner of the data does not have the control over who is allowed to access the data. The owner must trust the key-issuer who issues the appropriate keys to grant or deny access to the appropriate users.

To overcome this drawback, another form of ABE scheme was introduced in [7]. This scheme works in the reverse manner where the user keys are associated with sets of attributes and the ciphertexts are associated with the policies. Hence, it is referred to as the ciphertext-policy ABE (CP-ABE) scheme. The authors claim that their scheme is conceptually closer to role-based access control, but they do not provide a clear description as to how this can be achieved in terms of the characteristics associated with role-based access such as role hierarchies and constraints. Some variations of the CP-ABE schemes have been proposed such as in [8–10] with features such as chosen ciphertext attack (CCA) secure solution and constant size solution. However once again, these schemes have other disadvantages in practice such as the ability to achieve revocation. That is, revocation of a user's key will affect the keys possessed by all the other users, which makes such schemes inefficient in practice.

As an aside, in this paper, we will show how to enforce the RBAC policies using CP-ABE. First we associate the role with a policy of a set of attributes, and we say that a user belongs to the role if this user has the keys for all the attributes in the set for the role. To define a role A that inherits the permissions from another role B, we simply associate role A with a policy that contains the policy for role B. When the owner wants to encrypt a message to a role, he/she simply uses the CP-ABE scheme to encrypt the message under the policy of the role, and all the users in the role will be able to decrypt as they have the keys for all the attributes in the policy. The ABE scheme that we used in comparison refers to this approach.

Recently, another scheme has been proposed in [11] that considers a hierarchical role-based access control model. However, this scheme once again lacks the ability of user revocation, and the size of the ciphertext increases linearly with the number of all the predecessor roles.

The RBE scheme that we propose in this paper overcomes these limitations.

3. PRELIMINARIES

In this section, we first introduce the BE technique and the definitions of our RBE scheme. Then, we describe the security properties of our RBE scheme. Finally, we review the basic cryptographic principles that will be used throughout the paper.

3.1. Broadcast encryption

The concept of BE was introduced by Fiat and Naor [12]. In BE schemes, a broadcaster encrypts messages and transmits them to a group of users who are listening in a broadcast channel. Then they use their private keys to decrypt the transmissions. While encrypting the messages, the broadcaster can choose the set of users that is allowed to decrypt the messages. Following this original scheme, many other BE schemes have been proposed such as [13–15]. These schemes require public parameters for every user, and every time a user wants to join or leave the system, the public parameters need to be updated.

ID-based BE. In 1984, Shamir [16] suggested the possibility of a public key encryption scheme in which the public key can be an arbitrary string. In 2001, Boneh and Franklin introduced an ID-based encryption (IBE) scheme, in which the sender can use the identity of the receiver as the public key to encrypt the messages. An ID-based BE scheme (IBBE) is defined in a similar way. In an IBBE scheme, the system does not need to have any preset parameters for every user, and a broadcaster only needs to know the identity of the user if this user is allowed to decrypt the messages. In this case, one user joining or leaving the system will not affect any other user. Moreover, the users do not even need to have the decryption key at the time when the messages were encrypted. They can obtain their keys afterwards. Several IBBE schemes have been proposed subsequently in [17–19].

3.2. Formulation of RBE framework

Our RBE scheme consists of the following parties: a set of owners who store their private data in the cloud, a set of users $\mathcal{U}$ whom the owners may wish to share the private data with, a set of roles that these users can have, the associated role managers RM and a group administrator GA. In this paper, we will assume that there exists a single authority GA who has the authority to generate the keys for the users and the roles. In a subsequent paper, we will consider the extension of this scheme to multiple group administrators.

We define the following algorithms for our hybrid RBE scheme:

Setup (λ): takes as input the security parameter λ and outputs a master key mk and a group public key pk. mk is given to GA, and pk is made public.

CreateRole (mk, ID_R): an algorithm executed by the GA, on input of a role with its identity ID_R, generates the role secret

sk_R and returns a set of public parameters pub_R of the role and an empty user list $\mathcal{RUL}$ that will list all the users who are entitled to that role.

CreateUser (mk, ID_U): an algorithm on input of the identity ID_U of a user, executed by the GA, returns the decryption key dk to the user.

AddUser (pk, pub_R, $\mathcal{U}$, ID_U): an algorithm executed by the role manager RM, on the request of joining the role from the user, updates the role's public parameter pub_R and the user list $\mathcal{RUL}$ if the user qualifies the role.

Encrypt (pk, pub_R, M): an algorithm executed by the owner to encrypt the private data M outputs the ciphertext C to be stored in the cloud.

Decrypt (pk, pub_R, dk, C): an algorithm executed by the user to decrypt the ciphertext in the cloud outputs the plain text M if the user has the permission to access the data, and results in $\perp$ otherwise.

RevokeUser (pk, pub_R, $\mathcal{U}$, ID_U): an algorithm executed by the role manager RM and the GA, on input an identity ID_U of the user U, removes U from the $\mathcal{RUL}$ and updates the role's public parameters.

We assume that the cloud provider provides an underlying service to store the system public parameters and all the role public information. The GA is a trusted party who creates the roles and the keys for the users. We assume that the master secret mk is securely stored by the GA, and the GA can communicate with the Role Manager RM and users via a secure channel. When GA answers to the request *CreateUser*, it returns the generated information to the user via the secure channel.

The Role Manager (RM) is a trusted party per role and it manages the set of users in a given role. It can assign the role to a user if the user qualifies for the role, or exclude a user if the user is found to be malicious. When adding a user to a role, RM needs to verify the user's qualifications and determine whether the user can have this role. When a user resigns from a role or is removed from a role, RM revokes the role permissions for that user. We assume that there are some standard authentication mechanisms available that can be performed between the RM and a user. We will not consider these in this paper but they are being addressed as part of the implementation of the proposed scheme.

Users have the access to the ciphertext stored in the cloud. They are given appropriate decryption keys by the GA when they join the system. The users possess some credentials that are used to prove to an RM that they have appropriate qualifications to join a role. Users are able to use their decryption keys to decrypt the ciphertext, and we assume that the users are responsible for keeping their decryption key secure.

3.3. Security properties of RBE

Here are some of the important security properties that are achieved by our RBE scheme.

(i) Roles can be defined in a hierarchy, which means a role can have sub-roles (successor roles). The users in a role are able to decrypt the messages that are encrypted for its sub-roles, including any successor roles of its sub-roles.
(ii) The owner of the data can encrypt the private data to a specific role. Only the users in the specified role or predecessor roles are able to decrypt the data.
(iii) Users who are not entitled to the specified role or predecessor roles will not have the ability to decrypt and reveal the private data. Even the cloud provider (who stores the data) will not be able to see the content of the data if the provider is not given the appropriate role.
(iv) A user is able to join a role after the owner has encrypted the data for that role, and the user will be able to access that data from then on, and the owner does not need to re-encrypt the data. A user can be revoked at any time (e.g if he/she is found to be malicious), in which case the revoked user will not have access to any future data encrypted to this role.

Selective-ID Security: Our RBE scheme splits the encryption into two levels to avoid re-encryption when adding a new user into a role. Hence the security of our scheme is built on the standard notion of selective-ID security with an additional round. We say that our RBE scheme is semantically secure (IND-sID) if no polynomially bounded adversary $\mathcal{A}$ has a non-negligible advantage against the challenger in the following scenario:

Init: The adversary $\mathcal{A}$ first outputs a set $\{\mathsf{ID}_1, \ldots, \mathsf{ID}_k\}$ of identities that he wants to attack (with $k \leq q$).

Setup: The challenger takes a security parameter λ and runs the *Setup* algorithm. It gives the group public key pk to the adversary $\mathcal{A}$, and keeps the master key mk as secret.

Phase 1: The adversary $\mathcal{A}$ adaptively issues queries $q_1, \ldots, q_m$ where query q_i is one of:

(i) Create user query (ID_{U_i}). The challenger runs algorithm *CreateUser* to generate decryption key dk_i corresponding to the identity ID_{U_i}, and sends dk_i to the adversary $\mathcal{A}$.
(ii) Add user query ($\mathsf{ID}_{U_i}, \mathsf{ID}_R$). If ID_R has not been issued a query, the challenger first runs the algorithm *CreateRole* to generate the role public parameter Pub_R corresponding to the identity ID_R. Then the challenger runs algorithm *AddUser* on input of the identity ID_{U_i} to update the role public parameters Pub_R. It then sends the updated version of Pub_R to the adversary $\mathcal{A}$.

Challenge: When the adversary $\mathcal{A}$ decides that Phase 1 is over, it outputs two equal length plaintexts M_0, M_1 and user identities ID_U on which it wishes to be challenged, where ID_U did not appear in any query in Phase 1. Now the challenger runs the algorithm *AddUser* on input of the identity ID_U and a role public to update the role public parameters Pub_R. Then the challenger picks a random bit $b \in \{0, 1\}$, and runs the *Encrypt* algorithm to encrypt M_b and returns the ciphertext C and Pub_R to $\mathcal{A}$.

Phase 2: The adversary $\mathcal{A}$ adaptively issues more queries $q_{m+1}, \ldots, q_n$ where query q_i is one of:

(i) Create user query $(\mathsf{ID}_{U_i} | \mathsf{ID}_{U_i} \neq \mathsf{ID}_U)$. Challenger responds as in Phase 1.
(ii) Add user query $(\mathsf{ID}_{U_i}, \mathsf{ID}_R | \mathsf{ID}_{U_i} \neq \mathsf{ID}_U)$. Challenger responds as in Phase 1.

Guess: Finally, the adversary $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$, and wins the game if $b = b'$.

Revocable-ID Security: We define the selective Revocable-ID security of our scheme as follows. Since the revoked user has the ability to access the role secrets that were generated before the user has been revoked, we let the adversary $\mathcal{A}$ have the ability to query about the role secrets generated at any period of time except the one used at the challenge time. We still use the standard notion of selective-ID security, and define the sRID security of our scheme by the following scenario between an adversary $\mathcal{A}$ and a challenger.

Init: The adversary $\mathcal{A}$ first outputs a set $\{\mathsf{ID}_1, \ldots, \mathsf{ID}_k\}$ of identities that he wants to attack (with $k \leq q$).

Setup: The challenger takes a security parameter λ and runs the *Setup* algorithm. It gives the group public key pk to the adversary $\mathcal{A}$, and keeps the master key mk as secret.

Phase 1: The adversary $\mathcal{A}$ adaptively issues queries $q_1, \ldots, q_m$ where query q_i is one of:

(i) Create role query (ID_{R_i}). The challenger runs the algorithm *CreateRole* to generate role secret sk_{R_i} corresponding to the identity ID_{R_i}. Then it re-randomize the role secret and sends sk'_{R_i} to the adversary $\mathcal{A}$.

Challenge: When the adversary $\mathcal{A}$ decides that Phase 1 is over, it outputs two equal length plaintexts M_0, M_1 and an identity ID_R on which it wishes to be challenged, where ID_R did not appear in any query in Phase 1. Then the challenger picks a random bit $b \in \{0, 1\}$, and runs the *Encrypt* algorithm to encrypt M_b and returns the ciphertext C to $\mathcal{A}$.

Phase 2: The adversary $\mathcal{A}$ adaptively issues more queries $q_{m+1}, \ldots, q_n$, where query q_i is one of:

(i) Create role query (ID_{R_i}). Challenger responds as in Phase 1.

Guess: Finally, the adversary $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$, and wins the game if $b = b'$.

3.4. The bilinear pairings

Let $\mathbb{G}_1, \mathbb{G}_2$ be two cyclic additive groups of prime order p, and $\mathbb{G}_T$ be a cyclic multiplicative group of prime order p. g and h are two random generators where $g \in \mathbb{G}_1, h \in \mathbb{G}_2$. A bilinear pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ satisfies the following properties:

(i) *Bilinear:* for $a, b \in \mathbb{Z}_p^*$ we have $\hat{e}(g^a, h^b) = \hat{e}(g, h)^{ab}$.
(ii) *Non-degenerate:* $\hat{e}(g, h) \neq 1$ unless $g = 1$ or $h = 1$.
(iii) *Computable:* the pairing $\hat{e}(g, h)$ is computable in polynomial time.

3.5. Security assumptions

A General Decisional Diffie–Hellman Exponent (GDDHE) assumption has been given in [1], and it can be proved to have generic security with the GDHE assumption given in [20].

Let p be an integer prime and let s, n be positive integers. Let $P, Q \in \mathbb{F}_p[X_1, \ldots, X_n]^s$ be two s-tuples of n-variate polynomials over $\mathbb{F}_p$ and let $f \in \mathbb{F}_p[X_1, \ldots, X_n]$. We write $P = (p_1, p_2, \ldots, p_s)$ and $Q = (q_1, q_2, \ldots, q_s)$, and the first components of P, Q satisfy $p_1 = q_1 = 1$. For a set Ω, a function $h : \mathbb{F}_p \to \Omega$, and a vector $x_1, \ldots, x_n \in \mathbb{F}_p$, we write

$$h(P(x_1, \ldots, x_n)) = (h(p_1(x_1, \ldots, x_n)), \ldots, h(p_s(x_1, \ldots, x_n))) \in \Omega^s.$$

We use similar notation for the s-tuple Q. Let $\mathbb{G}, \mathbb{G}_T$ be groups of order p and let $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a non-degenerate bilinear map. Let $g \in \mathbb{G}$ be a generator of $\mathbb{G}_1$ and set $g_1 = e(g, g) \in G_T$.

We say that a polynomial $f \in \mathbb{F}_p[X_1, \ldots, X_n]$ is dependent on the sets (P, Q) if

$$f = \sum_{i,j=1}^{s} a_{i,j} p_i p_j + \sum_{i=1}^{s} b_i q_i, \quad \text{where } a_{i,j}, b_i \in \mathbb{Z}_p.$$

DEFINITION 3.1 (GDDHE Problem). *(P, Q, f)-GDDHE Problem in $\mathbb{G}$ is defined as follows: Given a random $T \in \mathbb{G}_T$ and the vector*

$$H(x_1, \ldots, x_n) = (g^{P(x_1,\ldots,x_n)}, g_1^{Q(x_1,\ldots,x_n)}) \in \mathbb{G}_1^s \times \mathbb{G}_T^s$$

decide whether $T = g^{f(x_1,\ldots,x_n)}$.

We use similar notation for the s-tuple R, and define a $(P, Q, R, f) - GDDHE$ assumption that extends $GDDHE$ Problem to the case with pairwise distinct roots. The security of our scheme is based on this assumption.

DEFINITION 3.2 ((P, Q, R, f)-GDDHE Problem). *Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e})$ be a bilinear map group system and let g and*

h be the generator of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, and $v = \hat{e}(g, h)$. Solving the $(P, Q, R, f) - GDDHE$ problem consists, given

$$g, g^{\gamma}, \ldots, g^{\gamma^{2n}}, h, h^{\gamma}, \ldots, h^{\gamma^{2n}},$$
$$(g^{P(x_1,\ldots,x_n)}, h^{Q(x_1,\ldots,x_n)}, v^{R(x_1,\ldots,x_n)})$$

and $K \in \mathbb{G}_T$, in deciding whether K is equal to $\hat{e}(g, h)^{f(x_1,\ldots,x_n)}$ or some random element of $\mathbb{G}_T$.

4. OUR RBE SCHEME

In this section, we propose our hybrid RBE scheme, and the scheme is designed as follows:

Setup (λ)**:** Generate three groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$, and an bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Choose random generators $g \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$, a random value $t \leftarrow \mathbb{Z}_p^*$, two secret values $s, k \leftarrow \mathbb{Z}_p^*$ and select two hash functions $H_1 : \{0, 1\}^* \to \mathbb{Z}_p^*$, $H_2 : \mathbb{G}_T \to \mathbb{Z}_p^*$. The master secret key mk and system public key pk are defined as

$$\mathsf{mk} = (s, k, g),$$
$$\mathsf{pk} = (w, v, y, g^k, h, h^s, \ldots, h^{s^q}),$$
$$\text{where } w = g^s, v = \hat{e}(g, h), y = g^t$$

and q is the maximum number of involved users and roles.

CreateRole $(\mathsf{mk}, \mathsf{ID}_R)$**:** To create a role with identity ID_R, where $\{\mathsf{ID}_{R_1}, \ldots, \mathsf{ID}_{R_m}\}$ are the identities of all the predecessor roles of role R, the role manager RM first outputs an empty role user list $\mathcal{RUL}$ and chooses a random secret value $r \leftarrow \mathbb{Z}_p^*$. Then RM computes

$$K = v^r, \quad W = w^{-r}$$

and sends K to GA via a secure channel. After receiving K, GA first generates the role secret sk_R

$$\mathsf{sk}_R = g^{\frac{1}{s+H_1(\mathsf{ID}_R)}}$$

and computes the role public parameters as

$$A = h^{(s+H_1(\mathsf{ID}_R))\prod_{i=1}^{m}(s+H_1(\mathsf{ID}_{R_i}))}, \quad B = A^k,$$
$$S = \mathsf{sk}_R \cdot y^k \cdot y^{H_2(K)} = g^{\frac{1}{s+H_1(\mathsf{ID}_R)}+t(k+H_2(K))}.$$

CreateUser $(\mathsf{mk}, \mathsf{ID}_U)$**:** When a user with identity ID_U is added to the system, GA simply computes the user secret

$$\mathsf{dk}_U = g^{1/(s+H_1(\mathsf{ID}_U))}$$

and gives dk_U to the user U.

AddUser $(\mathsf{pk}, \mathsf{pub}_{R_i}, \mathcal{N}, \mathsf{ID}_{U_k})$**:** A user U_k with identity ID_{U_k} wishes to join the role R_i, and assume that R_i already has a set $\mathcal{N}$ of n users, where $U_k \notin \mathcal{N}$. RM first computes

$$V_i = h^{r_i \cdot (s+H_1(\mathsf{ID}_{U_k}))\prod_{j=1}^{n}(s+H_1(\mathsf{ID}_{U_j}))}.$$

Then RM adds ID_{U_k} into $\mathcal{RUL}$ and outputs the role public information

$$(\mathsf{ID}_{R_i}, A_i, B_i, W_i, V_i, S_i, \mathcal{RUL}).$$

Encrypt $(\mathsf{pk}, \mathsf{pub}_{R_x}, M)$**:** Assume that the owner of the message $M \in \mathbb{G}_T$ wants to encrypt M for the role R_x. Given $\mathsf{pk} = (w, v, y, g^k, h, h^s, \ldots, h^{s^m})$, the owner randomly picks $z \leftarrow \mathbb{Z}_p^*$ and computes

$$C_1 = w^{-z}, \quad C_2 = y^{-z}, \quad C_3 = A_x{}^z, \quad C_4 = M \cdot v^{-z}.$$

The owner outputs the ciphertext

$$C = \langle C_1, C_2, C_3, C_4 \rangle.$$

Decrypt $(\mathsf{pk}, \mathsf{pub}_{R_i}, \mathsf{dk}_{U_k}, C)$**:** Assume role R_x has a set $\mathcal{R}$ of predecessor roles, and the set $\mathcal{M} = R_x \cup \mathcal{R}$ has m roles $\{R_1, \ldots, R_m\}$. $R_i \in \mathcal{R}$ is one predecessor role of R_x, and there is a set $\mathcal{N}$ of n users $\{U_1, \ldots, U_n\}$ in R_i, and the user $U_k \in \mathcal{N}$ who is entitled to the role R_i wants to decrypt the message M. U_k computes

$$M' = C_4 \cdot \left(\hat{e}(C_1, h^{p_{i,\mathcal{M}}(s)})\hat{e}(C_2, B_x)\hat{e} \times \left(\frac{S_i}{y^{H_2(K_i)}}, C_3\right)\right)^{1/\prod_{j=1,j\neq i}^{m} H_1(\mathsf{ID}_{R_j})},$$

where

$$K_i = (\hat{e}(\mathsf{dk}_{U_k}, V_i) \cdot \hat{e}(W_i, h^{p_{k,\mathcal{N}}(s)}))^{1/\prod_{j=1,j\neq k}^{n} H_1(\mathsf{ID}_{U_j})}$$

and

$$p_{i,\mathcal{M}}(s) = \frac{1}{s} \cdot \left(\prod_{j=1,j\neq i}^{m} (s + H_1(\mathsf{ID}_{R_j})) - \prod_{j=1,j\neq i}^{m} (H_1(\mathsf{ID}_{R_j})) \right),$$

$$p_{k,\mathcal{N}}(s) = \frac{1}{s} \cdot \left(\prod_{j=1,j\neq k}^{n} (s + H_1(\mathsf{ID}_{U_j})) - \prod_{j=1,j\neq k}^{n} (H_1(\mathsf{ID}_{U_j})) \right).$$

RevokeUser $(\mathsf{pk}, \mathsf{pub}_R, \mathcal{N}, \mathsf{ID}_U)$**:** To revoke a user U_k from a role R_i which has a set $\mathcal{N}$ of n users, and $U_k \in \mathcal{N}$. The role manager RM first removes ID_{U_k} from role user list $\mathcal{RUL}$ and chooses a random value $r_i' \leftarrow \mathbb{Z}_p^*$. Then RM computes

$$K_i' = v^{r_i'}, \quad W_i' = w^{-r_i'},$$
$$V_i' = h^{r_i' \cdot \prod_{j=1,j\neq k}^{n}(s+H_1(\mathsf{ID}_{U_j}))},$$

and sends K_i' to GA via a secure channel. Suppose $\{R_1, \ldots, R_m\}$ are all the existing roles in the system, GA then chooses a random secret value $t' \leftarrow \mathbb{Z}_p^*$, computes

$$y = g^{t'}, \quad S_j = g^{1/(s+H_1(\mathsf{ID}_{R_j}))+t'(k+H_2(K_j))}, \quad 1 \le j \le m$$

and publishes all these values.

The role public information now changes to

$$(\mathsf{ID}_{R_i}, A_i, B_i, W_i', V_i', S_i', \mathcal{RUL}).$$

Correctness: To check the correctness of the scheme, now we show that $M = M'$. Assume that dk_{U_k} is a valid decryption key

for identity ID_{U_k} of user U_k in the system, and U_k is the member of role ID_R which the message was encrypted to.

First, we look into the computation of K_i:

$$\begin{aligned} K_i^* &= \hat{e}(\mathsf{dk}_{U_k}, V_i) \cdot \hat{e}(W_i, h^{p_{k,\mathcal{N}}(s)}) \\ &= \hat{e}(g^{1/(s+H_1(\mathsf{ID}_{U_k}))}, h^{r_i \prod_{j=1}^{n}(s+H_1(\mathsf{ID}_{U_j}))}) \cdot \hat{e}(w^{-r_i}, h^{p_{k,\mathcal{N}}(s)}) \\ &= \hat{e}(g, h)^{r_i \prod_{j=1, j\neq k}^{n}(s+H_1(\mathsf{ID}_{U_j}))} \cdot \hat{e}(g, h)^{-r_i \cdot s \cdot p_{k,\mathcal{N}}(s)} \\ &= \hat{e}(g, h)^{r_i \cdot \prod_{j=1, j\neq k}^{n} H_1(\mathsf{ID}_{U_j})}. \end{aligned}$$

Then we have

$$K_i = (K_i^*)^{1/\prod_{j=1, j\neq k}^{n} H_1(\mathsf{ID}_{U_j})} = \hat{e}(g, h)^{r_i} = v^{r_i}.$$

Next we verify:

$$\begin{aligned} K^* &= \hat{e}(C_1, h^{p_{i,\mathcal{M}}(s)}) \cdot \hat{e}(C_2, B_x) \cdot \hat{e}(S_i \cdot y^{-H_2(K_i)}, C_3) \\ &= \hat{e}(g^{-zs}, h^{p_{i,\mathcal{M}}(s)}) \cdot \hat{e}(g^{-zt}, h^{k \prod_{j=1}^{m}(s+H_1(\mathsf{ID}_{R_j}))}) \\ &\quad \cdot \hat{e}(g^{1/(s+H_1(\mathsf{ID}_{R_i}))+kt}, h^{z \prod_{j=1}^{m}(s+H_1(\mathsf{ID}_{R_j}))}) \\ &= \hat{e}(g, h)^{-zs \cdot p_{i,\mathcal{M}}(s)+1/(s+H_1(\mathsf{ID}_{R_i})) \cdot z \prod_{j=1}^{m}(s+H_1(\mathsf{ID}_{R_j}))} \\ &= \hat{e}(g, h)^{z \cdot \prod_{j=1, j\neq i}^{m} H_1(\mathsf{ID}_{R_j})}. \end{aligned}$$

Thus, we have

$$M' = C_4 \cdot (K^*)^{1/\prod_{j=1, j\neq i}^{m} H_1(\mathsf{ID}_{R_j})} = M v^{-z} \cdot \hat{e}(g, h)^z = M.$$

In the earlier-mentioned scheme, the owner of the message is able to specify a role R and encrypts the message M to this role. Only this role R and the roles that are predecessors of the role R can decrypt the message with their role secrets or randomized role secrets. For an individual role, the randomized role secret is encrypted in such a way that only the users in the role have the ability to decrypt the randomized role secret, and therefore are able to recover the message M. Clearly a user who cannot decrypt the randomized role secret cannot learn anything about the content of the message.

Note that the encryption has been split into separate levels. The role manager can add new users without changing the randomized role secret, which means that new users will be able to decrypt the message that was encrypted before he/she was assigned to this role, and the owner does not need to re-encrypt the message. On the other hand, when revoking a user, a role manager will need to have the GA re-randomize the role secret, so that this user cannot use the previous randomized role secret to decrypt messages any more, but no other roles will be affected by the changes to this role.

Let us now illustrate the scheme using some simple example scenarios.

4.1. Example scenarios

In this section, we give two small examples to illustrate the proposed RBE scheme (see Fig. 1).

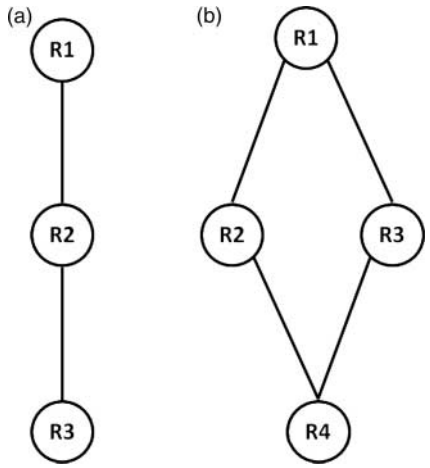

FIGURE 1. Hierarchical RBAC.

In Fig. 1a, the role R_1 is the predecessor role of role R_2, and the role R_2 is the predecessor role of role R_3. GA generates role public parameters pub_{R_i} for each role, where pub_{R_i} contain the identities of all the predecessor roles. When a user (owner of data) wishes to encrypt a message M for role R_3, pub_{R_3} will be used as the public key. Since pub_{R_3} includes the identities of R_1 and R_2, the message M is also encrypted for role R_1 and R_2.

When a user U_1 in role R_1 wants to decrypt the message, he/she only needs to use the user decryption key dk_{U_1} and the public parameters pub_{R_1} of role R_1 to decrypt the ciphertext. In this case, when the role manager wants to assign the role to a new user, RM only needs to update the pub_{R_1}.

Figure 1b illustrates a situation whereby the role R_1 is the predecessor role of two roles R_2 and R_3, and both R_2 and R_3 are predecessor roles of the role R_4. Firstly, GA generates role public parameters for each role. Note that pub_{R_4} will have the information for the identities of R_1, R_2 and R_3. Whenever a user encrypts a message M for role R_4, all the users in R_1, R_2 and R_3 will be able to decrypt the message. Here we can see that the size of pub_{R_4} remains constant regardless of the number of roles that inherit from it.

When adding a new role to the system, or removing an old role, only the successor roles will be affected. In Fig. 1b, if GA wants to remove R_3 from the hierarchical structure, only the pub_{R_4} will need to be updated, and R_1 and R_2 will not be affected.

Returning to our healthcare scenario, assume Alice is a patient who wants to store her health record in an online personal health record (PHR) cloud server, so that she can easily share the information with other people she chooses. Now let us assume that Alice wishes to allow her doctors to access her PHR but not the nurses in the practice. In our hybrid RBE scheme, we first create the roles for Doctor and Nurse, and each role public key contains an encrypt form randomized role secret that can

be used to decrypt the messages encrypted to the role itself. Alice simply encrypts her PHR to the role Doctor, and stores the ciphertexts in the online PHR cloud server. Assume that John is a doctor, and Jane is a nurse. Each of them will be given a decryption key.

Since Alice's PHR was encrypted to the role Doctor, John can use his decryption key with the public parameters of role Doctor to decrypt the randomized role secret of Doctor, and hence decrypt the Alice's PHR. Because Alice's PHR was not encrypted for the role Nurse, Jane cannot use role Nurse's public parameters to decrypt the ciphertexts; so she will learn nothing about Alice's PHR. Let us assume that Tom is one of the technicians who is maintaining the service of the cloud. Although Tom can access the ciphertexts stored in the cloud, he cannot use any role public parameters to decrypt, as he is not assigned to any role.

Now assume that the hospital has created a new role Staff, which is a successor role of role Doctor and Nurse. Let us assume that all the staff in the hospital will be added to this new role. The role Doctor and Nurse do not need to update their public parameters. If Alice has a message encrypted to the role Staff, John and Jane can both use their keys with the public parameters of role Doctor and Nurse to decrypt the Alice's message separately. When a new doctor joins the hospital, only the role Doctor needs to update the public parameters, and all the other roles will not be affected.

5. ANALYSIS OF OUR RBE SCHEME

5.1. Security analysis

In this section, we analyse our scheme to show that it is selective-ID secure(IND-sID) and revocable-ID secure(sRID).

THEOREM 5.1. *The proposed RBE scheme is IND-sID secure against active adversary under the GDDHE assumption in the Random Oracle Model.*

Suppose $\mathcal{A}$ is an attacker that wins the following game with probability $\mathsf{Adv}^{\mathcal{RBE}}_{\text{IND-sID}}$. Then we construct another attacker $\mathcal{B}$ that solves the *GDDHE* problem.

Let q be the maximum number of identities of users and roles that the adversary can query, $\mathcal{U} = \{\mathsf{ID}_{U_1}, \ldots, \mathsf{ID}_{U_n}\}$ and $\mathcal{R} = \{\mathsf{ID}_{R_1}, \ldots, \mathsf{ID}_{R_m}\}$ are the set of users' and roles' identities, respectively, that the adversary will issue the queries. $\mathcal{B}$ will be given $g_0, g_0^s, \ldots, g_0^{s^{2n}}, h_0, h_0^s, \ldots, h_0^{s^{2n}}$, where s is a random number chosen by the challenger. We define the following polynomials:

(i) $g_1(x,k) = \prod_{i=1}^{k}(x + H_1(\mathsf{ID}_{R_i}))$;
(ii) $g_2(x,k) = \prod_{i=1}^{k}(x + H_1(\mathsf{ID}_{U_i}))$;
(iii) $f(x) = g_1(x,m) \cdot g_2(x,n)$;
(iv) $f_1(x,i) = f(x)/(x + H_1(\mathsf{ID}^*_{R_i}))$, where $i \in [1,m]$;
(v) $f_2(x,i) = f(x)/(x + H_1(\mathsf{ID}^*_{U_i}))$, where $i \in [1,n]$.

Init: The adversary $\mathcal{A}$ first outputs a set $\{\mathsf{ID}_1, \ldots, \mathsf{ID}_k\}$ of identities that he wants to attack (with $k \le q$).

Setup: Adversary $\mathcal{B}$ will set the following values to generate the system parameters,

$$g = g_0^{f(s)}, \quad h = h_0, \quad w = g_0^{s \cdot f(s)},$$
$$v = \hat{e}(g,h) = \hat{e}(g_0,h_0)^{f(s)},$$
$$y = g^t = g_0^{t \cdot f(s)}, \quad \text{where } t \overset{R}{\leftarrow} \mathbb{Z}_q^*$$

Then $\mathcal{B}$ defines the public key as $\mathsf{pk} = (w, v, y, g^k, h, h^s, \ldots, h^{s^q})$, and creates the role parameters

$$A = h^{\prod_{i=1}^{m}(s+H_1(\mathrm{ID}_{R_i}))} = h_0^{g_1(s,m)},$$
$$B = h^{k\prod_{i=1}^{m}(s+H_1(\mathrm{ID}_{R_i}))} = h_0^{k \cdot g_1(s,m).}$$

Phase 1: The adversary $\mathcal{A}$ adaptively issues queries $q_1, \ldots, q_m$,

(i) If $\mathcal{A}$ has not issued the query on ID_{U_j}, $\mathcal{B}$ computes the user decryption key as

$$\mathsf{sk}_{U_j} = g^{1/(s+H_1(\mathsf{ID}_{U_j}))} = g_0^{f_2(s,j)}.$$

(ii) If $\mathcal{A}$ has issued the query on ID_{U_j}, $\mathcal{B}$ then computes the role public parameters for role with ID_{R_i} as follows to assign the role to the user, t is the number of the users currently in the role

$$K_i = v^{r_i}, \quad W_i = w^{-r_i} = g_0^{-r_i \cdot s \cdot f(s)},$$
$$V_i = h^{r_i \cdot \prod_{j=1}^{t}(s+H_1(\mathrm{ID}_{U_j}))} = h_0^{r_i \cdot g_1(s,t)},$$

and

$$S_i = y^{H_2(K_i)} \cdot g^{1/(s+H_1(\mathrm{ID}_{R_i}))} \cdot g^{kt}$$
$$= y^{H_2(K_i)} \cdot g^{f_1(s)+kt \cdot f(s).}$$

As we use the ID-based BE to encrypt the re-randomized role secret, and this IBBE scheme has been proven to be secure in [1], we assume this to be the case here as well; so the adversary will learn nothing about S_i here.

Challenge: Once $\mathcal{A}$ decides that Phase 1 is over, it publishes the identity ID_{R_i} of the role to which it wishes to encrypt the message and two messages M_0, M_1 on which it wishes to be challenged. Then $\mathcal{B}$ simulates *Encrypt* algorithm by constructing the ciphertext for the message M_b for a random $b \in \{0,1\}$ as,

$$C_1 = w^{-z} = g_0^{-z \cdot s \cdot f(s)}, \quad C_2 = y^{-z} = g_0^{-z \cdot t \cdot f(s)},$$
$$C_3 = A^z = h_0^{z \cdot g_1(s,m)}, \quad C_4 = M_b \cdot \hat{e}(g_0,h_0)^{-z \cdot f(s)}.$$

Phase 2: The adversary $\mathcal{A}$ adaptively issues more queries $q_{m+1}, \ldots, q_n$, which does the same as the steps in **Phase 1**.

Guess: Finally, the adversary $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$, and wins the game if $b = b'$.

Then we have

$$\begin{aligned}\mathsf{Adv}^{\text{gddhe}} &= \frac{1}{2}(\Pr[b = b'|\text{real}] + \Pr[b = b'|\text{rand}]) - \frac{1}{2} \\ &= \frac{1}{2}\left(\frac{1}{2} + \mathsf{Adv}^{\mathcal{RBE}}_{\text{IND-sID}}\right) + \frac{1}{2}\cdot\frac{1}{2} - \frac{1}{2} \\ &= \frac{1}{2}\cdot\mathsf{Adv}^{\mathcal{RBE}}_{\text{IND-sID}}.\end{aligned}$$

THEOREM 5.2. *Our RBE scheme is selective Revocable-ID secure against active adversary under the GDDHE assumption in the Random Oracle Model.*

Suppose $\mathcal{A}$ is an attacker that wins the following game with probability $\mathsf{Adv}^{\mathcal{RBE}}_{\text{sRID}}$. Then we construct another attacker $\mathcal{B}$ that solves the *GDDHE* problem.

Let q be the maximum number of identities of roles that the adversary can query, $\mathcal{R} = \{\mathsf{ID}_{R_1}, \ldots, \mathsf{ID}_{R_m}\}$ is the set of roles' identities that the adversary will issue the queries. $\mathcal{B}$ will be given $g_0, g_0^s, \ldots, g_0^{s^{2n}}, h_0, h_0^s, \ldots, h_0^{s^{2n}}$, where s is a random number chosen by the challenger. We define the following polynomials:

(i) $f(x) = \prod_{i=1}^{m}(x + H_1(\mathsf{ID}_{R_i}))$;
(ii) $g(x, i) = f(x)/(x + H_1(\mathsf{ID}_{R_i}))$, where $i \in [1, m]$.

Init: The adversary $\mathcal{A}$ first outputs a set $\{\mathsf{ID}_1, \ldots, \mathsf{ID}_k\}$ of identities that he wants to attack (with $k \leq q$).

Setup: Adversary $\mathcal{B}$ will set the following values to generate the system parameters:

$$\begin{aligned}&g = g_0^{f(s)}, \quad h = h_0, \quad w = g_0^{s\cdot f(s)}, \\ &v = \hat{e}(g, h) = \hat{e}(g_0, h_0)^{f(s)}, \\ &y = g^t = g_0^{t\cdot f(s)}, \quad \text{where } t \overset{R}{\leftarrow} \mathbb{Z}_q^*.\end{aligned}$$

Then $\mathcal{B}$ defines the public key as $\mathsf{pk} = (w, v, y, g^k, h, h^s, \ldots, h^{s^q})$, and creates the role parameters

$$\begin{aligned}A &= h^{\prod_{i=1}^{m}(s+H_1(\mathrm{ID}_{R_i}))} = h_0^{f(s)}, \\ B &= h^{k\prod_{i=1}^{m}(s+H_1(\mathrm{ID}_{R_i}))} = h_0^{k\cdot f(s)}.\end{aligned}$$

Phase 1: The adversary $\mathcal{A}$ adaptively issues queries $q_1, \ldots, q_m$,

(i) If $\mathcal{A}$ has not issued the query on ID_{R_i}, $\mathcal{B}$ computes the role secret and re-randomizes as

$$\mathsf{sk}_{R_i} = g^{1/(s+H_1(\mathsf{ID}_{U_i}))+kt} = g_0^{g(s,i)+kt\cdot f(s)}.$$

(ii) If $\mathcal{A}$ has issued the query on ID_{R_i}, $\mathcal{B}$ then re-randomizes the role secret as

$$\mathsf{sk}_{R_i} = g^{1/(s+H_1(\mathsf{ID}_{U_i}))+kt'} = g_0^{g(s,i)+kt'\cdot f(s)}$$

and update the system public parameter y to

$$y = g^{t'} = g_0^{t'\cdot f(s)}.$$

Challenge: Once $\mathcal{A}$ decides that Phase 1 is over, it publishes the identity ID_{R_i} of the role to which it wishes to encrypt the message and two messages M_0, M_1 on which it wishes to be challenged. Then $\mathcal{B}$ simulates *Encrypt* algorithm by constructing the ciphertext for the message M_b for a random $b \in \{0, 1\}$ as,

$$\begin{aligned}&C_1 = w^{-z} = g_0^{-z\cdot s\cdot f(s)}, \quad C_2 = y^{-z} = g_0^{-z\cdot t\cdot f(s)}, \\ &C_3 = A^z = h_0^{z\cdot f(s)}, \quad C_4 = M_b\cdot\hat{e}(g_0, h_0)^{-z\cdot f(s)}.\end{aligned}$$

Phase 2: The adversary $\mathcal{A}$ adaptively issues more queries $q_{m+1}, \ldots, q_n$, which does the same as the steps in **Phase 1**.

Guess: Finally, the adversary $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$, and wins the game if $b = b'$.

Then we have

$$\begin{aligned}\mathsf{Adv}^{\text{gddhe}} &= \frac{1}{2}(\Pr[b = b'|\text{real}] + \Pr[b = b'|\text{rand}]) - \frac{1}{2} \\ &= \frac{1}{2}\left(\frac{1}{2} + \mathsf{Adv}^{\mathcal{RBE}}_{\text{sRID}}\right) + \frac{1}{2}\cdot\frac{1}{2} - \frac{1}{2} \\ &= \frac{1}{2}\cdot\mathsf{Adv}^{\mathcal{RBE}}_{\text{sRID}}.\end{aligned}$$

Chosen-Ciphertext Security: Our security definitions and proofs are in the chosen-plaintext model. However, our scheme can be extended to the chosen-ciphertext model by using a standard transformation in [21]. Let $\mathcal{E}$ be a probabilistic public key encryption scheme, and $\mathcal{E}_{pk}(M; r)$ be the encryption of M using the random bits r under the public key pk. A hybrid scheme $\mathcal{E}^{hy}$ is defined in [21] as

$$\mathcal{E}^{hy}_{pk}(M) = \mathcal{E}_{pk}(\delta; H(\delta, M))\|G(\delta) \oplus M,$$

where H and G are two hash functions $H : \mathsf{MSPC}\times\{0, 1\}^n \to \{0, 1\}^n$, $G : \mathsf{MSPC} \to \{0, 1\}^n$ and δ is randomly chosen from MSPC. Fujisaki and Okamoto [21] show that if $\mathcal{E}$ is a one-way encryption scheme, then $\mathcal{E}^{hy}$ is a chosen ciphertext secure system (IND-CCA) in the random oracle model. Since semantic security implies one-way encryption, this result also applies if $\mathcal{E}$ is semantically secure (IND-sID-CPA).

Now we apply this generic conversion to our proposed CCA secure RBE scheme and the chosen-ciphertext secure RBE scheme is as follows:

Setup (λ)**:** Same as in the original RBE scheme with two additional hash functions $H_3 : \mathbb{G}_T\times\{0, 1\}^n \to \mathbb{Z}_p^*$, $H_4 : \mathbb{G}_T \to \{0, 1\}^n$.

CreateRole ($\mathsf{mk}, \mathsf{ID}_R$)**:** Same as in the original RBE scheme.

CreateUser ($\mathsf{mk}, \mathsf{ID}_U$)**:** Same as in the original RBE scheme.

AddUser ($\mathsf{pk}, \mathsf{sk}_{R_i}, \mathcal{N}, \mathsf{ID}_{U_k}$)**:** Same as in the original RBE scheme.

Encrypt ($\mathsf{pk}, \mathsf{pub}_R, M$)**:** Assume that the owner of the message $M \in \{0,1\}^n$ wants to encrypt M for the role R_x. Given $\mathsf{pk} = (w, v, y, g^k, h, h^s, \ldots, h^{s^m})$, the owner randomly picks $\delta \leftarrow \mathbb{G}_T$, sets $z = H_3(\delta, M)$ and computes

$$C_1 = w^{-z}, \quad C_2 = y^{-z}, \quad C_3 = A_x{}^z, \quad C_4 = \delta \cdot v^{-z}.$$

The owner outputs the ciphertext

$$C = \langle C_1, C_2, C_3, C_4, H_4(\delta) \oplus M \rangle.$$

Decrypt ($\mathsf{pk}, \mathsf{pub}_R, \mathsf{dk}_{U_k}, C$)**:** Let $C = \langle C_1, C_2, C_3, C_4, C_5 \rangle$ be a ciphertext. The user first computes δ as in the original scheme, and then computes

$$M = C_5 \oplus H_4(\delta)$$

RevokeUser ($\mathsf{pk}, \mathsf{sk}_R, \mathcal{N}, \mathsf{ID}_U$)**:** Same as in the original RBE scheme.

5.2. Scheme efficiency

Our scheme has several distinct advantages compared with the schemes that are earlier mentioned in the Related Work in Section 2.

5.2.1. *Size of ciphertext and decryption key*

Our scheme has constant-size ciphertext and decryption key. That is, the size of the ciphertext and decryption key is independent of the number of roles and users in the system, and it will not increase when new roles are created or more users join the system. These are significant features when it comes to development of large-scale systems. In comparison, for the schemes given in [2, 7], the size of the ciphertext is linearly proportional to the number of users, which make these schemes inefficient in practical systems where there can be a large number of users. Furthermore, in these schemes, when a user has multiple roles, multiple copies of the keys or ciphertexts are required for a single user in every role, while in our scheme, no matter how many roles that a user have, the size of keys and ciphertexts remain constant. The scheme in [11] has reduced the size of the ciphertext to be linearly proportional to the depth of the role hierarchy. However their scheme is not able to deal with user revocation, and the ciphertext size is still not constant.

5.2.2. *User management*

The schemes mentioned in the Related Work in Section 2 all assume the existence of a single trusted party who manages the role memberships of the users. In this case, this single trusted party needs to verify the qualification of the user whenever a role wishes to add a new user. However, in the real world, whether a user is entitled to a role or not is usually decided by different entities who are responsible for validating the users' qualifications for different roles. In our scheme, the role memberships of the users are no longer decided by a single trusted party. Instead, they are controlled by role managers, who could be different for each role. This is more natural and reflects the situation in practice. A role manager can add or revoke a user without having to gain admissions from other parties, including the trusted party.

As mentioned earlier, in our scheme, a role manager RM can exclude the user from accessing future encrypted data. In [11], once the user obtains the decryption key, the manager cannot revoke the user's permission even if the user does not qualify for the role, as the decryption relies on the hierarchical structure of the roles. Unless the predecessor roles have changed, the user can always decrypt the messages. In addition, when a user has been excluded in our scheme, the other users of the same role will not be affected. In the scheme described in [2], as the key structure had been constructed based on the access matrix, when a user is removed from the access matrix, the key management structure needs to be updated as well. This in turn will change the keys of all the other users. In the scheme [7], as the ciphertext is associated with the hierarchical structure, the system parameters need to be re-generated every time a user's permission is revoked, which results in all the other users having to update their keys. In our scheme, when a user is excluded, we can see that only the roles need to update their public information.

TABLE 1. Comparison of schemes.

	Our scheme	[2]	[7]	[11]
Ciphertext length	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(m)$
Encryption complexity	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
Decryption complexity	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
Computation round(s) in decryption	1	m	m	1
Users revocation	Yes	Yes	Yes	No
Users affected in a revocation	0	n	n	

5.2.3. *Computations*

Table 1 shows a comparison of our scheme with the other schemes in [2, 7, 11]. The Table assumes that there are n users of the same role and there are m roles between the role of the user and the specified role for the encryption in the role hierarchical structure.

The decryption algorithm in our scheme only requires one round of computation, while in [2, 7], the user needs to compute the secrets of all the roles (nodes) that are on the path from the entitled role to the target one in the policy tree.

As shown in Table 1, though the scheme in [2] is computationally more efficient in terms of decryption, the complexity of role creation, user grant and revocation operations is linearly proportional to the number of authorized users. This makes the scheme not scalable in practice. Moreover, the number of computation rounds in decryption is linearly proportional to the number of roles in between in the role hierarchical structure. However our scheme, with its constant size ciphertext and decryption key, is more efficient for large scale systems, and the decryption only requires one computation round. Comparing our scheme to the other two schemes in [7, 11], our scheme has similar performance but has constant size ciphertext and decryption key. Furthermore, our scheme also supports dynamic user revocation while the other two schemes do not. Effective dynamic user revocation is a fundamental requirement in most practical systems.

6. DISCUSSION

In this section, we briefly consider some aspects that can be optimized in the implementation of the proposed RBE scheme.

6.1. Preventing malicious cloud provider

When an owner of the data wants to encrypt private data, he/she needs to obtain the system public parameters $w, v \in \mathsf{pk}$ and role public information $A_R \in \mathsf{pub}_R$ from the cloud in order to run the *Encrypt* algorithm. This process can be susceptible to a man-in-the-middle attack because a malicious cloud provider can return the pub'_R of another role R' instead of the real pub_R of the role R that the owner wants to encrypt the private data with. If this is done, then the malicious cloud provider will be able to reveal the content of data without the knowledge of the owner.

In order to prevent this kind of attack, we propose the following: first we make the GA return system parameters w, v together with the decryption key to the user while executing the *CreateUser* algorithm. Then we let GA sign the value A_R under the identity of role R using an appropriate ID-based signature scheme (e.g. [22, 23]), and return the signature $\mathsf{sig}(A_R, \mathsf{ID}_R)$ along with A_R. Upon receiving A_R in running the *Encrypt* algorithm, the owner should verify the signature corresponding to the identity of the role, which he/she wants to encrypt the private data to, before sending the ciphertext to the cloud. If the verification fails, it means that the value A_R is not the public information of the specified role R and the owner should not use it to encrypt the data.

6.2. Decrypting previously generated messages

After executing the *RevokeUser* algorithm in our RBE scheme, the role public information pub_R is updated to prevent the excluded user from decrypting any future messages. However, the *Decrypt* algorithm uses pub_R, which means the messages encrypted before executing *RevokeUser* algorithm cannot be decrypted even by the existing users of the role because the pub_R used to decrypt the previously generated messages has been replaced.

We can allow for this and enable decryption of previously generated messages in our RBE scheme by simply creating the indices for the ciphertext C and role public information pub_R. When RM runs the *AddUser* or *RevokeUser* algorithm to create or update pub_R, the algorithm will create a unique index for pub_R, and insert it in the cloud rather than replacing the previous pub_R. When an owner of the data is encrypting the private data, the *Encrypt* algorithm can be modified to attach the index of the latest version pub_R to the ciphertext C.

When a user wants to decrypt a ciphertext C, he/she can ask the cloud provider to return the ciphertext C and pub_R of the role. The cloud provider can easily choose and return the corresponding pub_R according to the index included in the ciphertext C.

Moreover, the role public information's update in *RevokeUser* algorithm requires the changes in the public parameter S of all the roles in the system, and it could be inefficient when there is a large number of roles. However, this work can be delegated to a trust authority by GA without giving out the master secret. Such an approach is being addressed as part of the implementation of the scheme.

6.3. Optimizing decryption strategy

In our RBE scheme, let us assume that m is the number of predecessor roles of a specific role, and assume that there are n users in the same role with the user who runs the *Decrypt* algorithm. The *Decrypt* algorithm requires the expansion of two polynomials of $m+1$ and $n+1$ degrees, respectively. This calculation could be time-consuming if m and n are very large numbers.

We note that these two polynomials remain the same in the decryption of two different messages if the identities of roles and users are not changed. Therefore, in the implementation, the user can keep these values as auxiliary information to help with decrypting messages. These values only need to be re-calculated when the predecessor roles of the specific role are changed or the permission is revoked from another user in the same role.

In addition, the user needs the system public $\mathsf{pk} = (w, v, g^k, h, h^s, \ldots, h^{s^q})$ to calculate the expansion of the polynomials, which is inefficient in practice because downloading the pk every time could cost network traffic if q is a very large number. We can then utilize the computing power of the cloud to do the polynomial expansion because calculating them does not require any secret values, and the cloud only needs to return the result of the polynomial expansion to the user, which can markedly simplify the work for the user as well.

More precisely, in the decryption algorithm, the user in role R_1 wants to decrypt the message that is encrypted to role R_2, and

R_1 inherits all the permissions of R_2. When the user retrieves the ciphertext from the cloud, the cloud computes the value of $h^{p_{i,\mathcal{M}}(s)}$ and $h^{p_{k,\mathcal{N}}(s)}$ for the user and returns them to the user. This user can keep two auxiliary values $(\text{Aux}_R, \text{Aux}_U)$, where

$$\text{Aux}_R = h^{p_{i,\mathcal{M}}(s)}, \quad \text{Aux}_U = H_2(K_i).$$

If a user wants to decrypt other messages that are encrypted to role R_2 in future, Aux_R can be reused in *Decrypt* algorithm, and user only needs to ask the cloud to re-calculate the value when one or more predecessor roles of R_2 are changed. Similarly, user can reuse Aux_U to decrypt any future messages, and Aux_U only needs to be re-calculated when one or more users are excluded from R_1. This 'caching' strategy can improve the decryption performance substantially as the user only needs to compute three pairings most of the time.

6.4. Implementation issues

In this subsection, we briefly discuss some issues related to the implementation.

After the users have been given the decryption keys (when they join the system), the users should take the responsibility to store their secret keys safely. If a user's secret key is revealed to others, other parties will have the ability to decrypt any data on behalf of the actual user. If a user discovers that his/her secret key has been revealed to other users, he/she should report this to all appropriate role managers immediately and request that this user's identity be excluded from the role.

Before a user is included into a role, the role manager will need to authenticate the user so that the role manager can be convinced that the user qualifies for the role. We have not considered the authentication mechanisms in this paper; we have assumed that such mechanisms exist and assume that the role manager will assign the role only to the qualified users in the system.

In our scheme, the cloud platform is only used for storage purposes. The encryption and decryption computations do not occur in the cloud; hence the original private data from the owner and the secret key of a user will not be given to the cloud. Even if the cloud may be involved in the decryption algorithm, the computations in the cloud will not involve any secret values from either the owners or the users.

In our RBE scheme, the system public pk is stored publicly in the cloud, and can be retrieved by any user, who has access to the cloud, to encrypt the data. However, in the case where a user needs to identify the original source of the data, we can simply employ an ID-based digital signature scheme (e.g. [22, 23]), and have the owner sign the ciphertext. Then the ciphertext can be stored along with its signature. When a user wants to decrypt the ciphertext, he/she will need to verify the signature to check if the ciphertext is correctly signed by the owner from whom the user is expecting the data.

7. CONCLUDING REMARKS

In this paper, we have considered security requirements for storage of information in the cloud and proposed a hybrid RBE scheme that combines role-based access control with encryption to address them. We have constructed a specific RBE scheme using the BE scheme described in [1]. We have conducted security analysis of our scheme and have given proofs to show that our scheme is secure against adaptive attack and revocable-ID attack. We have discussed the performance and efficiency of our scheme and have compared it with other previously related work. We have shown that our scheme has several superior characteristics such as constant size ciphertext and decryption key, efficient user revocation and user management, and the ability to handle role hierarchies. We have also considered some aspects that can be optimized to achieve efficient implementation. We believe that the proposed scheme is suitable for large scale systems, especially in the context of achieving user-centric secure information storage in a cloud computing environment. To provide further administrative convenience and scalability, we are currently developing an administrative model for our RBE scheme.

REFERENCES

[1] Delerablée, C. (2007) Identity-based Broadcast Encryption with Constant Size Ciphertexts and Private Keys. *ASIACRYPT*, Kuching, Malaysia, December 2–6, pp. 200–215. Springer, Berlin.

[2] Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S. and Samarati, P. (2007) Over-encryption: Management of Access Control Evolution on Outsourced Data. *VLDB*, University of Vienna, Austria, September 23–27, pp. 123–134. ACM, New York.

[3] Zych, A., Petkovic, M. and Jonker, W. (2008) Efficient key management for cryptographically enforced access control. *Comput. Stand. Interfaces*, **30**, 410–417.

[4] Blundo, C., Cimato, S., di Vimercati, S.D.C., Santis, A.D., Foresti, S., Paraboschi, S. and Samarati, P. (2009) Efficient Key Management for Enforcing Access Control in Outsourced Scenarios. *SEC*, Pafos, Cyprus, May 18–20, pp. 364–375. Springer, Berlin.

[5] Atallah, M.J., Frikken, K.B. and Blanton, M. (2005) Dynamic and Efficient Key Management for Access Hierarchies. *ACM Conf. Computer and Communications Security*, Alexandria, VA, USA, November 7–11, pp. 190–202. ACM, New York.

[6] Goyal, V., Pandey, O., Sahai, A. and Waters, B. (2006) Attribute-based Encryption for Fine-grained Access Control of Encrypted Data. *ACM Conf. Computer and Communications Security*, Alexandria, VA, USA, October 30–November 3, pp. 89–98. ACM, New York.

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

[8] Cheung, L. and Newport, C.C. (2007) Provably Secure Ciphertext Policy Abe. *ACM Conf. Computer and Communications Security*, Alexandria, VA, USA, October 28–31, pp. 456–465. ACM, New York.

[9] Ibraimi, L., Tang, Q., Hartel, P.H. and Jonker, W. (2009) Efficient and Provable Secure Ciphertext-policy Attribute-based Encryption Schemes. *ISPEC*, Xi'an, China, April 13–15, pp. 1–12. Springer, Berlin.

[10] Emura, K., Miyaji, A., Nomura, A., Omote, K. and Soshi, M. (2009) A Ciphertext-policy Attribute-based Encryption Scheme with Constant Ciphertext Length. *ISPEC*, Xi'an, China, April 13–15, Lecture Notes in Computer Science 5451, pp. 13–23. Springer, Berlin.

[11] Zhu, Y., Ahn, G.-J., Hu, H. and Wang, H. (2010) Cryptographic Role-based Security Mechanisms Based on Role-key Hierarchy. *ASIACCS*, Beijing, China, April 13–16, pp. 314–319. ACM, New York.

[12] Fiat, A. and Naor, M. (1993) Broadcast Encryption. *CRYPTO*, Santa Barbara, CA, USA, August 22–26, pp. 480–491. Springer, New York.

[13] Garay, J.A., Staddon, J. and Wool, A. (2000) Long-lived Broadcast Encryption. *CRYPTO*, Santa Barbara, CA, USA, August 20–24, pp. 333–352. Springer, New York.

[14] Halevy, D. and Shamir, A. (2002) The LSD Broadcast Encryption Scheme. *CRYPTO*, Santa Barbara, CA, USA, August 18–22, pp. 47–60. Springer, New York.

[15] Boneh, D., Gentry, C. and Waters, B. (2005) Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys. *CRYPTO*, Santa Barbara, CA, USA, August 14–18, pp. 258–275. Springer, New York.

[16] Shamir, A. (1985) Identity-based Cryptosystems and Signature Schemes. *Proc. CRYPTO 84 on Advances in Cryptology*, Santa Barbara, CA, USA, August 19–22, pp. 47–53. Springer, New York.

[17] Delerablée, C., Paillier, P. and Pointcheval, D. (2007) Fully Collusion Secure Dynamic Broadcast Encryption with Constant-Size Ciphertexts or Decryption Keys. *Pairing*, Tokyo, Japan, July 2–4, pp. 39–59. Springer, Berlin.

[18] Boneh, D. and Hamburg, M. (2008) Generalized Identity Based and Broadcast Encryption Schemes. *ASIACRYPT*, Melbourne, Australia, December 7–11, pp. 455–470. Springer, Berlin.

[19] Hu, L., Liu, Z., and Cheng, X. (2010) Efficient identity-based broadcast encryption without random oracles. *J. Comput.*, **5**, 331–336.

[20] Boneh, D., Boyen, X. and Goh, E.-J. (2005) Hierarchical Identity Based Encryption with Constant Size Ciphertext. *EUROCRYPT*, Aarhus, Denmark, May 22–26, pp. 440–456. Springer, Berlin.

[21] Fujisaki, E. and Okamoto, T. (1999) Secure Integration of Asymmetric and Symmetric Encryption Schemes. *CRYPTO*, Santa Barbara, CA, USA, August 15–19, pp. 537–554. Springer, New York.

[22] Hess, F. (2002) Efficient Identity Based Signature Schemes Based on Pairings. *Selected Areas in Cryptography*, St. John's, Newfoundland, Canada, August 15–16, pp. 310–324. Springer, Berlin.

[23] Cha, J.C. and Cheon, J.H. (2003) An Identity-based Signature from Gap Diffie–Hellman Groups. *Public Key Cryptography*, Miami, FL, USA, January 6–8, pp. 18–30. Springer, Berlin.