Cryptographic Role-based Security Mechanisms Based on Role-Key Hierarchy *

Yan Zhu[†‡] Gail-Joon Ahn[§] Hongxin Hu[§] Huaixi Wang[♭]

[†]Institute of Computer Science and Technology, Peking University, Beijing 100871, China
[‡]Key Laboratory of Network and Software Security Assurance (Peking University), Ministry of Education, China
[§]Laboratory of Security Engineering for Future Computing (SEFCOM), Arizona State University, Tempe, AZ 85287, USA
[♭]School of Mathematical Sciences, Peking University, Beijing 100871, China
{yan.zhu,wanghuaixi}@pku.edu.cn; {gahn,hxhu}@asu.edu

ABSTRACT

Even though role-based access control (RBAC) can tremendously help us minimize the complexity in administering users, it is still needed to realize the notion of roles at the resource level. In this paper, we propose a practical cryptographic RBAC model, called role-key hierarchy model, to support various security features including signature, identification and encryption based on role-key hierarchy. With the help of rich algebraic structure of elliptic curve, we introduce a role-based cryptosystem construction to verify the rationality and validity of our proposed model. Also, a proof-of-concept prototype implementation and performance evaluation are discussed to demonstrate the feasibility and efficiency of our mechanisms.

Keywords

Access Control, Role-based Cryptosystem, Role-Key Hierarchy, Pairing-based Cryptosystem

1. BACKGROUND AND MOTIVATION

Role-based access control (RBAC), as a proven alternative to traditional access control including discretionary access control (DAC) and mandatory access control (MAC), has been widely adopted for various information systems over the past few years [14]. Even though RBAC can tremendously help us minimize the complexity in administering users, it is still needed to realize the notion of roles at the resource level. In other words, RBAC systems need to control a user's access to resources as well as resource-level management based on roles. Consequently, in order to provide effective resource management, it is inevitable to adopt various cryptographic capabilities for managing resources in RBAC systems. However, the existing cryptographic schemes based on common asymmetric cryptosystem have several limitations to address above-mentioned features since those schemes cannot accommodate access control features of RBAC and have a lack of scalability and interoperability due to inconsistent parameters among cryptographic mechanisms.

In distributed environments, we can leverage RBAC models to enforce fine-grained policies for sharing resources [10]. However, the current cryptosystems do not support such shared modes because the encryption/decryption keys cannot be recognized between RBAC systems. As a consequence, the resources should be re-encrypted when they are transferred into another domain. Obviously, it is necessary to design an efficient cryptographic mechanism compatible with corresponding access control systems.

In fact, the research for cryptographic hierarchical structure has a long history since hierarchical structure is a nature way to organize and manage a large number of users. Several approaches on cryptographic partial order relation supporting hierarchical structure have been proposed. Akl and Taylor introduced a simple scheme to solve multilevel security problem [1, 2]. Since then, several efficient methods have been studied. The concept of Logical Key Hierarchy (LKH) was proposed by Wallner et al. [16] and Wong et al. [17]. In this paradigm, common encryption key was organized into a tree structure to achieve secure group communication in a multicast environment. Additionally, public-key hierarchy cryptosystems have been recently proposed. For instance, hierarchical identity-based encryption (HIBE) mirrors an organizational hierarchy [8]. Although the public key can be an arbitrary multi-level string, the HIBE schemes support tree structures and provide an efficient method to assign a subset of users to encrypt the message. Another important area is hierarchy key management (HKM) that also organizes the key into a hierarchy. For example, time-bound hierarchical key assignment (THKA) [15] can assign time-dependent encryption keys to a set of classes in a partially ordered hierarchy. This scheme is especially suited for the realtime broadcast system with time control. Unfortunately, these existing schemes cannot manage each user's key and thus all users with same identity (or security level) share the same key. In other words, all users are indiscriminate for various operations in systems. Therefore, the existing

*Supported by 863 Project of China (No.2006AA01Z434) and NSF of China (No. 10990011). An extended abstract of this paper appears in ACM ASIACCS 2010, pages 314-319, ACM, 2010. Online as IACR ePrint Report 2010/188

schemes are hard to realize some advanced security functions such as revocation, digital forensics, undeniability and traceability.

Some new technologies, such as identity-based encryption (IBE) [5], attribute-based encryption (ABE) [11], and public-key broadcast encryption (PBE) [7], lay out a solid foundation for designing an efficient cryptosystem. Inspired by these techniques, in this paper, we propose a practical cryptographic RBAC model, called role-key hierarchy model, to support a variety of security features including signature, identification and encryption based on role-key hierarchy. With the help of rich algebraic structure of elliptic curve, we introduce a role-based cryptosystem construction to verify the rationality and validity of our proposed model. This constructions can provide more efficient and flexible control than other hierarchical key assignments [3]. More importantly, some unique security mechanisms, such as role-based signature & authentication and role-based encryption are supported by our construction.

The rest of the paper is organized as follows. Section 2 overviews the role hierarchy in RBAC and Section 3 articulates our role-key hierarchy structure along with the usability of this structure in Section 4. In Section 5, we address our RBC construction and application schemes in depth. In Section 6, we evaluate the security and performance of our schemes. Finally, we conclude this paper with our future work.

2. PRELIMINARIES

2.1 Partial Orders

Let $\Psi = \langle P, \preceq \rangle$ be a (finite) partially ordered set with partial order relation $\preceq$ on a (finite) set P. A partial order is a reflexive, transitive and anti-symmetric binary relation. Inheritance is reflexive because a role inherits its own permissions, transitivity is a natural requirement in this context, and anti-symmetry rules out roles that inherit from one another and would therefore be redundant.

Two distinct elements x and y in Ψ are said to be comparable if $x \preceq y$ or $y \preceq x$. Otherwise, they are *incomparable*, denoted by $x \| y$. An order relation $\preceq$ on P gives rise to a relation $\prec$ of strict inequality: $x \prec y$ in P if and only if (or iff) $x \preceq y$ and $x \neq y$. Also, if x is dominated by y, we denote the domination relation as $x \prec_d y$. In addition, if $x \prec y$ and $x \preceq z \prec y$, it then implies $z = x$. The latter condition demands that there be no element z of P satisfying $x \prec z \prec y$. We define the predecessors and successors of elements in $\Psi = \langle P, \preceq \rangle$ as follows: For an element x in P, $\uparrow x = \{y \in P | x \preceq y\}$ denotes the set of predecessors of x, $\downarrow x = \{y \in P | y \preceq x\}$ denotes the set of successors.

2.2 Role Hierarchy

In an information system, a hierarchy is used to denote the relationships and arrangements of the objects, users, elements, values, and so on. Especially, in many access control systems the users are organized in a hierarchy constructed with a number of classes, called security classes or roles, according to their competencies and responsibilities. This hierarchy arises from the fact that some users have more access rights than others.

In order to manage large-scale systems, the hierarchy in RBAC becomes more complex than other systems. Especially, role hierarchy (RH) is a natural means for structuring roles to reflect an organization's lines of authority and responsibility. We adopt the definitions from RBAC models proposed by Sandu et al. [13]:

Definition 1. [Hierarchical RBAC model]*: The RBAC model has the following components:*

- *U, R, P, and S, users, roles, permissions and sessions respectively,*
- *$PA \subseteq P \times R$, a many-to-many permission to role assignment relation.*
- *$UA \subseteq U \times R$, a many-to-many user to role assignment relation.*
- *$RH \subseteq R \times R$ is a partial order on R called the role hierarchy or role dominance relation, written as $\preceq$,*
- *$user : S \to U$, a function mapping each session s_i to the single user $user(s_i)$, and*
- *$roles : S \to 2^R$, a function mapping each session s_i to a set of roles: $roles(s_i) \subseteq \{r \in R | \exists r' \in R, r \preceq r' : (user(s_i), r') \in UA\}$ and s_i has the permissions: $\bigcup_{r \in roles(s_i)} \{p \in P | \exists r'' \in R, r'' \preceq r : (p, r'') \in PA\}$.*

A hierarchy in RBAC is mathematically a partial order that defines an inheritance (or seniority) relation between roles, whereby senior roles acquire the permissions of their juniors. An example of role hierarchy is shown in Figure 1, in which more powerful (senior) roles are shown toward the top of the diagram and less powerful (junior) roles toward the bottom.

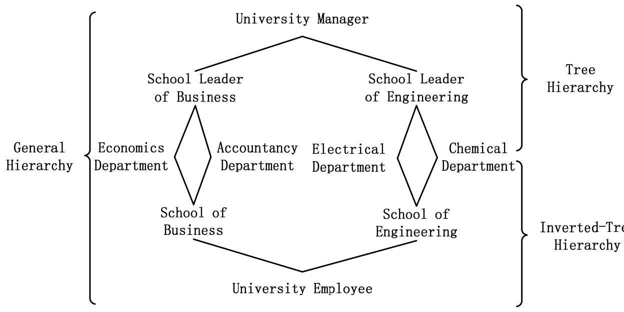

Figure 1: Example of role hierarchy with tree, inverted-tree, and general hierarchies.

Based on the specific features of resource management, we divide role hierarchy into three categories:

1. *Tree hierarchy*: It is useful to support the **sharing** of resources, in which resources make available to junior roles are also available to senior roles.
2. *Inverted-tree hierarchy*: It allows the **aggregation** of resources from more than one role, in which the senior can access resources in all subordinate roles.
3. *General hierarchy*: It can compose various different structures into a role hierarchy. Thus it facilitates both the **sharing and aggregation** of resources.

3. ROLE KEY HIERARCHY

3.1 Role-Key Hierarchy Structure

In order to incorporate cryptographic schemes with RBAC, we propose a new hierarchy structure called **Role-Key Hierarchy** (RKH). Based on the hierarchical RBAC model, we define RKH as follows:

Definition 2. [Role-Key Hierarchy]*: Given a role hierarchy* $\langle R, \preceq \rangle$ *in RBAC, role-key hierarchy is a cryptographic partial order relation for the sets of users, keys, and roles, denoted by* $\mathcal{H} = \langle U, K, R, \preceq \rangle$*, satisfying the following conditions:*

1. $K = PK \cup SK$*, the key set* K *includes the role-key set* PK *and the user-key set* SK*;*
2. $UKA \subseteq U \times SK$*, a one-to-one user to key assignment relation, i.e., each user* $u_{i,j} \in U$ *is assigned to an exclusive* user-key $sk_{i,j} \in SK$*;*
3. $RKA \subseteq R \times PK$*, a one-to-one role to key assignment relation, i.e., each role* $r_i \in R$ *corresponds to a unique* role-key $pk_i \in PK$*;*
4. $KH \subseteq PK \times PK$*, is a partial order on* PK *called the key hierarchy or key dominance relation, also written as* $\preceq$*; and*
5. *Each user* $u_{i,j}$ *can access the resources associated with* r_l *if and only if* $r_l \preceq r_i \in RH$ *and* $(u_{i,j}, r_i) \in UA$*.*

where, $\langle K, \preceq \rangle$ *is the smallest partially ordered set satisfying the above conditions. The user holds multiple user keys if he is member of multiple roles in role hierarchy.*

In RBAC systems, various access control functions are designated by permissions P. In the same way, the RBAC permissions can be designated by some cryptographical algorithms, such as $Encrypt$ and $Decrypt$, which can realize various access control functions by using role keys and user keys in role-key hierarchy. These algorithms can also be used independently to protect files from unauthorized access while these resources break away from the scope of this RBAC systems or an attacker gains physical access to the computer.

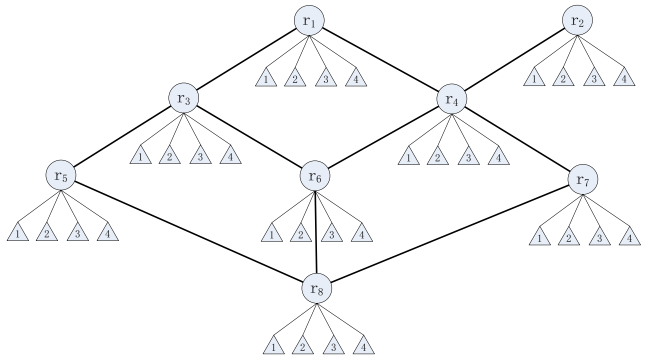

Figure 2: Example of role-key hierarchy

Our main objective is to map the role hierarchy in RBAC into a key management system. According to the condition 3 and 4, the role key set PK should have the same structure as the role hierarchy structure. Moreover, each user key $sk_{i,j} \in SK$ also needs to contain necessary information about role hierarchy for dealing with access functions independently by itself. Figure 2 shows an example of role-key hierarchy, in which the circle denotes the role key and the triangle denotes the user key, respectively. Note that this is a general hierarchy.

3.2 Role-based Cryptosystem

For ease of use, we expect that a system manager assigns the user key $sk_{i,j} = (lab_{i,j}, dk_{i,j})$ to a user, where $lab_{i,j}$ is a public label and $dk_{i,j}$ is a private key. This label $lab_{i,j}$ can be used to realize special functions such as designation, revocation, and tracing.

Given a role hierarchy $\Psi = \langle R, \preceq \rangle$ and a security parameter s, **Role-based Cryptosystem** (RBC) is a key management system that can construct a role-key hierarchy $\mathcal{H} = \langle U, K, R, \preceq \rangle$ on Ψ and generate all keys on $\mathcal{H}$, which is specified by three randomized algorithms, $Setup$, $KeyRGen$, and $AddUser$, described as follows:

- *Setup*(s, Ψ): Takes a security parameter s and a role hierarchy Ψ as an input. It produces a manager key mk and an initial parameter $params$, that is, $Setup(s, \Psi) \rightarrow \{\mathcal{H}, mk, params\}$.
- *GenRKey*$(params, r_i)$*:* Takes the parameter $params$ and a role index r_i. It generates a role key pk_i in r_i, that is, $KeyRGen(params, r_i) \rightarrow pk_i$.
- *AddUser*$(mk, ID, u_{i,j})$*:* Takes a user identity ID, a user index $u_{i,j}$, and the manager key mk. It outputs a user secret key, which involves a user label $lab_{i,j}$ and a private key $dk_{i,j}$, for the user $u_{i,j}$, that is, $AddUser(mk, ID, u_{i,j}) \rightarrow sk_{i,j} = (lab_{i,j}, dk_{i,j})$. The user label $lab_{i,j}$ is added to the public encryption key: $params = params \bigcup \{lab_{i,j}\}$.

In public-key settings, a user does not hold any private information and the permission process is performed only with the help of the public role key $\{pk_i\}$ containing the user's labels $\{lab_{i,j}\}$, which is also called as ID-based RBC because the user's public labels can be used to support the various functions.

3.3 Security Goal of RKH

Obviously, security requirements in general cryptosystem are not sufficient enough to reflect the requirements of role-key hierarchy. It is important to consider typical attacks when we try to design key hierarchy and its schemes. In contrast with existing key hierarchy, RKH has several unique features:

1. Each user $u_{i,j}$ is assigned to an exclusive user key $sk_{i,j}$, by which certain users can be chosen or identified in the processes of encryption, revocation, and tracing;
2. Public-key cryptography can be introduced to ensure the security of a user's private key even if the role key makes public in some systems. Therefore the role keys can be stored anywhere by RBAC systems; and
3. The *derivation* function of a user's private key is forbidden even for the cases of partial order relations,
$$\Pr[Delegate(sk_{i,j}, c_l) = sk_{l,j'}] \leq \epsilon, \forall c_l \preceq c_i. \quad (1)$$
where, ϵ is small enough. Hence a user cannot use this capability to obtain new keys or identities.

In order to ensure system security, RKH also needs to satisfy following properties:

- Each user in a role cannot get permissions to access another role's objects except for its subordinates, Also, a user cannot forge other's secret keys;
- The role key can be modified to satisfy the requirements of constraint policy, but it should not interfere with the issued keys of others; and
- To support the capability of audit capability, there exists an efficient tracing algorithm to identify the corrupted users or gain the corresponding evidence.

The RKH is a group-oriented cryptography with "1:n" character, where one role key corresponds to many user keys. Hence, in addition to passive cryptanalysis, the collusion attack is a major attack, which focuses on changing the privilege of the granted users or getting the other users' keys. This kind of attack involves the following cases:

- Collusion attack for framing users, in which the corrupted users in $\mathcal{R} = \{u_{i_k,j_k}\}_{k=1}^{t}$ wish to forge a new or unused key in $U \setminus \mathcal{R}$ (called as honest user). The aim of this attack is to avoid tracing and frame innocent users.
- Collusion attack for role's privilege, in which the corrupted users in $R = \{u_{i_k,j_k}\}_{k=1}^{t}$ wish to forge a new or unused key in $R \backslash \{r_{i_1}, \cdots, r_{i_t}\}$. The aim of this attack is to change the privilege in partial order hierarchy.

We also present a formal security model for two cases of collusion attacks in Appendix A. It is a challenging task to avoid collusion attack since the traitors (corrupted users) have been granted users before they are detected. Traitor tracing is an efficient method to tackle this attack. However, we must ensure that the traitors cannot forge an 'unused' key to avoid tracing but leave some 'foregone' clue of evidence to discover them.

The number of colluders $|\mathcal{R}| = t$ is an important parameter. A RBC scheme is to be (t, n, m)-collusion secure if for any subset of t in $\mathcal{R}$ with $|U| = n$ and $|R| = m$, the adversary can gain the advantage from $\mathcal{R}$ to break this scheme. It is said to be fully collusion secure when it is (n, n, m)-collusion secure.

4. CRYPTOGRAPHIC SECURITY MECHANISMS BASED ON RKH

4.1 Role-based Signature

The signature is a mathematical scheme for demonstrating the authenticity of a digital message or document. In RBAC model, the roles assigned to a user can be considered as one kind of identities of the user. Hence, a user could use his own roles to sign a resource. In other words, such a signature scheme provides a method to allow a user to *anonymously* sign a message on behalf of his roles. We call it **Role-based Signature** (RBS). The formal definition of RBS is provided as follows:

DEFINITION 3 (ROLE-BASED SIGNATURE). *A role-based signature scheme is a digital signature consisted of the following four procedures:*

Initial: *Takes role hierarchy $\langle R, \preceq \rangle$, and returns the role-key hierarchy $\mathcal{H} = \langle U, K, R, \preceq \rangle$ according to Setup and GenRKey algorithms in RBC model;*

Sign: *Takes the role-key pk_i for r_i, a user key $sk_{i,j}$, and a message $M \in \{0,1\}^*$, and returns a signature σ: $Sign(pk_i, sk_{i,j}, M) \rightarrow \sigma$;*

Verify: *Takes the role-key pk_i and a purported signature σ on a message M. It returns the validation result which would be either valid or invalid. The latter response can mean either that σ is not a valid signature, or that the user who generated has been revoked (in a set of revoked users, RL): $Verify(pk_i, \sigma, M) \rightarrow valid/invalid$;*

Trace: *Takes a user key $sk_{i,j}$ then this algorithm can trace a signature σ to at least one role member $u_{i,j}$ who generated it: $Trace(sk_{i,j}, \sigma) \rightarrow valid/invalid$.*

The trace algorithm allows a third party to undo the signature anonymity using a special trapdoor and recognize the original signer. A secure role-based signature scheme must satisfy following properties:

- Correctness: This requires that, for all $K = (PK, SK)$ generated by role-key hierarchy, valid signatures by role members can always be verified correctly, and invalid signatures should fail in the verification phase:
$$Verify(pk_i, Sign(pk_i, sk_{i,j}, M), M) = valid. \quad (2)$$
- Unforgeability: Only members of a role can create valid signatures with the role.
- Anonymity: Given a message and its signature, the identity of the individual signer cannot be determined without the manager key mk.
- Traceability: Given any valid signature, the manager or trusted third party should be able to trace who issued the signature by the user's secret key.
- Unlinkability: Given two messages and their signatures, we cannot determine whether the signatures were from the same signer or not.

In autonomous systems, role-based signature is used to verify the legality of the source of input data transmitted from other hosts or devices. This is more important for information sharing systems to prevent harmful information flows.

4.2 Role-based Authentication

Authentication allows access control systems to gain sufficient assurance that the identity of certain entity is legitimate as claimed. Cryptography-based authentication is widely adopted in current systems because it provides a higher level of security than password-based authentication. In addition, a real-time authentication for high-risk operations is necessary to prevent a user from changing roles after logging in. The authentication on RBAC should support two qualitative classes of identifications:

- User-based authentication, which is used to validate a user's identity, but the systems need to store the user's role information; and

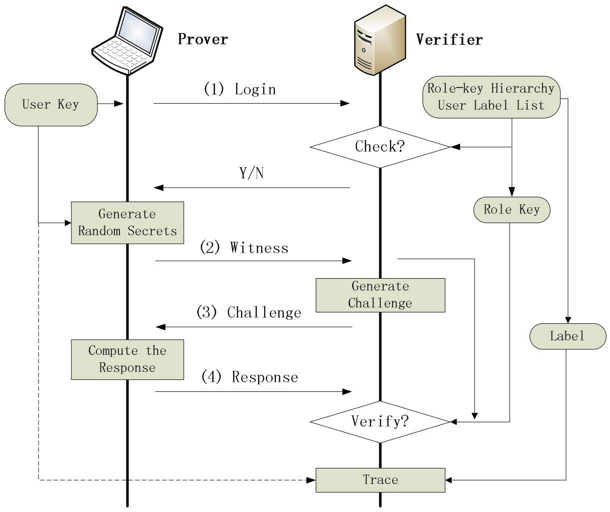

Figure 3: Authentication protocol based on RBC.

- Role-based authentication, which can provide identifiable evidences that a given user possesses the attributes of a given role.

Obviously, role-based authentication is a useful way for anonymous accesses, sharing systems, or off-line devices while the user information (including the user's public key in PKI) is not maintained by themselves. Furthermore, this approach can help achieve the interoperability as well. Hence, we propose a common framework of **Role-based Authentication** (RBA) based on a challenge-response protocol as shown in Figure 3.

DEFINITION 4 (ROLE-BASED AUTHENTICATION). *A role-based authentication scheme is a challenge-response identification protocol between prover (P) and verifier (V), consisting of following four procedures:*

Initial: *Takes role hierarchy $\langle R, \preceq \rangle$, and returns the role-key hierarchy $\mathcal{H} = \langle U, K, R, \preceq \rangle$ according to Setup and GenRKey algorithms in RBC model;*

Interact: *the prover and the verifier execute the protocol:*

1. **Login:** *The prover sends the label of identity (including rolename and username) to the verifier, then the verifier checks the availability by searching user-label database or role hierarchy: $P \rightarrow V : r_i \vee lab_{i,j}$;*
2. **Witness:** *If the check succeeds, the verifier requires the prover to return the witness of the verifier's private key on a random number r: $P \rightarrow V : S = Oneway(sk_{i,j}, r)$;*
3. **Challenge:** *The verifier selects a challenge (random number) and sends it to the prover: $P \leftarrow V : c = Random()$;*
4. **Response:** *After receiving the challenge c, the prover computes the response in terms of his private key $sk_{i,j}$ and the random numbers r in the witness, and sends it back to the verifier: $P \rightarrow V : s = Respose(sk_{i,j}, r, c)$.*

Verify: *The verifier verifies whether the response is consistent with the commitment, the challenge, and the role key: $Verify(pk_i, S, c, s) \rightarrow valid/invalid$. In the case of user-based authentication, he can also check the validity of the prover's label: $Verify(pk_i, S, c, s, lab_{i,j}) \rightarrow valid/invalid$.*

Trace: *Takes a prover key $sk_{i,j}$ then it can analyze an existing record $re = \langle S, c, s, r_i \rangle$ to verify whether or not this prover generated this record: $Trace(sk_{i,j}, re) \rightarrow valid/invalid$.*

Similarly to role-based signature, role-based authentication protocols must satisfy the following properties: correctness, anonymity, traceability, and unlinkability. Moreover, the following attacks should be avoided:

- Impersonation: a deception whereby one entity purports to be another;
- Replay attack: an impersonation or other deception involving information from a previous protocol execution on the same or a different verifier;
- Interleaving attack: an impersonation or other deception involving selective combination of information from one or more previous or ongoing protocol executions; and
- Chosen-text attack: an adversary strategically chooses challenges in an attempt to extract information about the manager key.

4.3 Role-based Encryption

Encryption systems allow users to encrypt resources (files or data) on disk, or synchronously transfer messages among multiple systems. Many encryption file systems have been developed in Windows and Linux environments, e.g., Windows Encrypting File System (EFS), SiRiUS [9] and Plutus [12]. However, these systems implement some trivial schemes where the number of ciphertexts in the file header grows linearly with the increased number of users who have permissions to access the file. To overcome such a limitation, we introduce a new scheme called **Role-based Encryption** (RBE), which can be used to improve the performance of existing encryption file systems.

DEFINITION 5 (ROLE-BASED ENCRYPTION). *A role-based encryption scheme is an encryption system consisting of the following three procedures:*

Initial: *Takes role hierarchy $\langle R, \preceq \rangle$, and returns the role-key hierarchy $\mathcal{H} = \langle U, K, R, \preceq \rangle$ according to Setup and GenKey algorithms in RHC model;*

Encrypt: *Takes the encryption key pk_i and a plaintext M. It produces a ciphertext $\mathcal{C}$: $Encrypt(pk_i, M) \rightarrow \mathcal{C}_i$.*

Decrypt: *Takes the user key $sk_{i,j}$ and the ciphertext $\mathcal{C}$. It generates the plaintext M: $Decrypt(sk_{i,j}, \mathcal{C}_l) \rightarrow M$, where $r_l \preceq r_i$.*

The relationship between encryption and decryption can be described as follows:

$$Decrypt(sk_{i,j}, Encrypt(pk_l, M)) = M \tag{3}$$

where $r_l \preceq r_i$ and $(u_{i,j}, r_i) \in UA$.

In order to improve the performance, we assume the following encrypted file structure: A file M is stored in the form $\langle Hdr_{pk_l}(S, ek), E_{ek}(M)\rangle$, where ek is a session key for encrypting M via a symmetric encryption method E, and S denotes the subset of the authorized users or the access control policy of this file. A user in S can use his private keys to decrypt the session key ek from $Hdr_{pk_l}(S, ek)$ and then decrypt the file M from $E_{ek}(M)$. The file system based on this structure is also called as role-based encryption file system (R-EFS) with revocation.

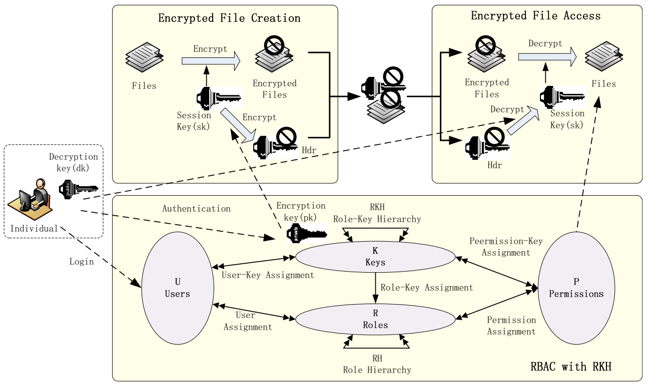

Figure 4: Role-based encryption file system.

Figure 4 illustrates a role-based encryption file system constructed based on role-key hierarchy, where each role r_i is assigned to an encryption key pk_i and each user has a few decryption keys $\{sk_{i,j}\}$. An administrator only needs to keep the manager key mk, but the pk_i could be saved in the public directory of the system. When a user $u_{i,j}$ in r_i wants to create an encrypted file, the RBAC systems encrypt the file with a session key ek, then encrypt ek by using the user's pk_i. The result is placed in the file header Hdr after the user gets the permissions from the RBAC systems. The user can also allow an arbitrary subset of the authorized users to decrypt the file by performing proper assignments or making the access control policy into S if necessary. When a user $u_{i,j}$ wants to access an encrypted file, the session key ek is recovered by $sk_{i,j}$, and the RBAC systems check whether $u_{i,j} \in S$ or the access control policy in S is valid. The file is decrypted by ek after the user gets the permissions from the RBAC systems.

This scheme can provide following security features for file systems:

1. Protection against data leakage on the physical device, possibly caused by an untrusted administrator, a stolen laptop or a compromised server;

2. Detection and prevention of unauthorized data modifications using a syncretic security mechanism based on policy-based access control and dynamic cryptographic technology;

3. Changing users' access privileges by dynamically converging the information of users' decryption keys to generate one-time role-based encryption keys; and

4. Enabling better scalability because all users are organized into a uniformed role-based cryptographic framework. Most of cryptographic operations are performed at role level rather than at user level.

5. PROPOSED SCHEMES

In this section, we present our role-based cryptosystem scheme with role-key hierarchy based on pairing-based cryptosystem. Meanwhile, role-based signature & authentication and role-based encryption mechanisms are addressed based on the proposed role-based cryptosystem construction.

5.1 Bilinear Pairings

We set up our systems using bilinear pairings proposed by Boneh and Franklin [4, 6]. Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be three cyclic groups of large prime order p. $\mathbb{G}_1$ and $\mathbb{G}_2$ are two additive group and $\mathbb{G}_T$ is a multiplicative group using elliptic curve conventions. Let $\hat{e}$ be a computable bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$[1] with the following properties: For any $G \in \mathbb{G}_1$, $H \in \mathbb{G}_2$ and all $a, b \in \mathbb{Z}_p$, we have

1. Bilinearity: $e([a]G, [b]H) = e(G, H)^{ab}$.

2. Non-degeneracy: $e(G, H) \neq 1$ unless G or $H = 1$.

3. Computability: $e(G, H)$ is efficiently computable.

Where, $[a]P$ denotes the multiplication of a point P in elliptic curve by a scalar $a \in \mathbb{Z}_p$. A bilinear map group system $\mathbb{S}$ is a tuple $\mathbb{S} = \langle p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e\rangle$ composed of the objects as described above. $\mathbb{S}$ may also include group generators in its description.

5.2 Role-based Cryptosystem Scheme

Let $\mathcal{H} = \{U, K, R, \preceq\}$ is a role-key hierarchy with partial-order $\preceq$. Without loss of generality, we assume that the total number of roles is m in $\mathcal{H}$, i.e., $R = \{r_1, r_2, \cdots, r_m\}$. We construct a RBC scheme as follows:

- $Setup(s, \Psi)$: Let $\mathbb{S} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ be a bilinear map group system with randomly selected generators $G \in \mathbb{G}_1$ and $H \in \mathbb{G}_2$, where $\mathbb{G}_1$ and $\mathbb{G}_2$ be bilinear group of prime order p. This algorithm first picks a random integer $\tau_i \in \mathbb{Z}_p^*$ for each role r_i in role-key hierarchy graph.[2] We define

$$\begin{cases} U_i &= [\tau_i]G \in \mathbb{G}_1 \quad \forall r_i \in R, \\ V &= e(G, H) \qquad \in \mathbb{G}_T. \end{cases} \tag{4}$$

  Each τ_i is called as the secret of a role and U_i is the identity of a role. Further, it defines $U_0 = [\tau_0]G$ by using a random $\tau_0 \in \mathbb{Z}_p^*$. Thus, public parameter is

$$params = \langle H, V, U_0, U_1, \cdots, U_c\rangle \tag{5}$$

  and we keep $mk = \langle G, \tau_0, \tau_1, \cdots, \tau_m\rangle$ secret.

- $GenRKey(params, r_i)$: This is an assignment algorithm for role encryption key from the setup parameter pp. For a role r_i, the role key pk_i can be computed as follows:

$$\begin{cases} pk_i &= \langle H, V, W_i, \{U_k\}_{r_k \in \uparrow r_i}\rangle \\ W_i &= U_0 + \sum_{r_i \not\preceq r_k} U_k, \end{cases} \tag{6}$$

[1]We require that no efficient isomorphism $\mathbb{G}_2 \rightarrow \mathbb{G}_1$ or $\mathbb{G}_1 \rightarrow \mathbb{G}_2$ is known, or $\mathbb{G}_2 \rightarrow \mathbb{G}_1$ is known but its inverted $\mathbb{G}_1 \rightarrow \mathbb{G}_2$ is unknown.

[2]Since the total number of roles is far less than the size of space of keys, we can use an efficient method to avoid the collision of value of role keys, e.g., the fast sort algorithm can be use to search the collision.

where, $\{U_k\}_{r_k \in \uparrow r_i}$ is the set of all roles in $\uparrow r_i$, which denotes the control domain for the role r_i. It is clear that $W_i = \left[\tau_0 + \sum_{r_i \not\preceq r_k} \tau_k\right] G$. For sake of simplicity, let $\zeta_i = \tau_0 + \sum_{r_i \not\preceq r_k} \tau_k$, so that we have $W_i = [\zeta_i]G$.

- $AddUser(mk, ID, u_{i,j})$: Given $mk = \langle G, \{\tau_i\}_{i=0}^{m} \rangle$ and a user index $u_{i,j}$ in the role r_i, the manager generates a unique decryption key by randomly selecting a fresh $x_{i,j} = Hash(ID, u_{i,j}) \in \mathbb{Z}_p^*$ and defining $dk_{i,j} = \langle A_{i,j}, B_{i,j} \rangle$ where

$$\begin{cases} lab_{i,j} &= & x_{i,j} & \in \mathbb{Z}_p^* \\ A_{i,j} &= & \left[\frac{x_{i,j}}{\zeta_i + x_{i,j}}\right] G & \in \mathbb{G}_1, \\ B_{i,j} &= & \left[\frac{1}{\zeta_i + x_{i,j}}\right] H & \in \mathbb{G}_2. \end{cases} \quad (7)$$

  Note that, the total number of users is unlimited in each role.

Finally, the above process outputs the set of role keys $\{pk_i\}$ and the set of user keys $\{sk_{i,j}\}$. More importantly, the security of user keys is not compromised even though role keys are available in public.

Let us now turn to the problem of validity. We know that two arbitrary roles have one of three relations: $r_i \preceq r_j$, $r_j \preceq r_i$, and $r_i || r_j$, so that partial order relation in role keys can be defined as

$$\sum_{r_i \not\preceq r_k} U_k = \sum_{r_k \in Ind(r_i)} U_k + \sum_{r_k \in Succ(r_i)} U_k, \quad (8)$$

where, $Ind(r_i)$ and $Succ(r_i)$ denote the set of incomparable roles and successors for r_i, respectively. This is illustrated in Figure 5 (the top is senior-most roles and the bottom is junior-most roles), with the key representation of W_i on the left of the node and U_i on the right. We first prove that this assignment works as required:

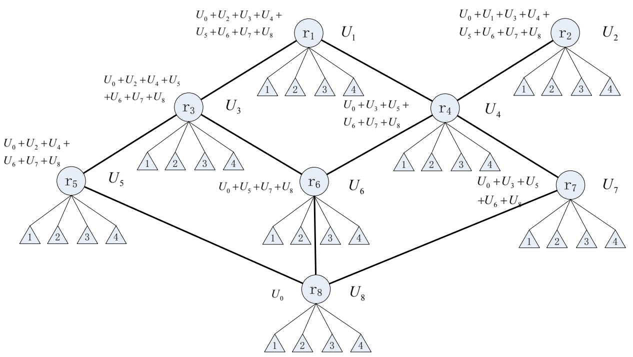

Figure 5: Example of role-key relationship on RBC.

THEOREM 1. *Under the above assignment, $\cup_{r_j \not\preceq r_k}\{\tau_k\} \subset \cup_{r_i \not\preceq r_k}\{\tau_k\}$ if and only if $r_j \prec r_i$.*

PROOF. The proof is immediate. First if $r_j \prec r_i$, then we have $r_i \in \cup_{r_j \preceq r_k}\{r_k\}$, so that $\cup_{r_i \preceq r_k}\{r_k\} \subset \cup_{r_j \preceq r_k}\{r_k\}$. It is easy to see that $\cup_{r_j \not\preceq r_k}\{r_k\} \subset \cup_{r_i \not\preceq r_k}\{r_k\}$ still holds. In terms of the corresponding relation between r_i and τ_i, we have $\cup_{r_j \not\preceq r_k}\{\tau_k\} \subset \cup_{r_i \not\preceq r_k}\{\tau_k\}$. Conversely, if $\cup_{r_j \not\preceq r_k}\{\tau_k\} \subset \cup_{r_i \not\preceq r_k}\{\tau_k\}$, then we know $\cup_{r_j \not\preceq r_k}\{r_k\} \subset \cup_{r_i \not\preceq r_k}\{r_k\}$. This relation can become $\cup_{r_i \preceq r_k}\{r_k\} \subset \cup_{r_j \preceq r_k}\{r_k\}$. Since $r_i \in \cup_{r_i \preceq r_k}\{r_k\}$, we have $r_j \prec r_i$. Hence, the theorem holds. $\square$

Due to U_i is chosen at random, this scheme do not permit the collision among the role keys, i.e., $pk_i = pk_j$ for $i \neq j$. The following theorem tells us that this collision is neglectable only if the security parameter s is large enough, moreover, the fast sort algorithm can help us to find the collision.

THEOREM 2. *The collision probability among n integers, which are chosen from p integers at random, is less than $\frac{(n+1)^2}{4p}$.*

PROOF. Firstly, the collision probability between t random integers $\{a_i\}_{i=1}^{t}$ and r random integers $\{b_i\}_{i=1}^{r}$, $\sum_{i=1}^{t} a_i = \sum_{i=1}^{r} b_i$, is $\frac{1}{p}$, where $a_1, \cdots, a_t \in_R \mathbb{Z}_p^*$ and $b_1, \cdots, b_r \in \mathbb{Z}_p^*$. Secondly, the number of all possible cases of $t+r=k$ is $\lfloor \frac{k}{2} \rfloor$ for $1 \leq t, r < n$, such that the number of all possible cases of $3 \leq t+r \leq n$ is $\sum_{k=3}^{n} \lfloor \frac{k}{2} \rfloor < \sum_{k=1}^{n} \frac{k}{2} = \frac{n(n+1)}{4} < \frac{(n+1)^2}{4}$. Hence, in terms of Bernoulli's inequality, the collision probability is $1 - (1 - \frac{1}{p})^{\frac{(n+1)^2}{4}} \leq \frac{(n+1)^2}{4} \frac{1}{p} = \frac{(n+1)^2}{4p}$. Note that we do not assume that a_i and b_j are different for $1 \leq i \leq t$ and $1 \leq j \leq r$. $\square$

Since the total number of roles is far less than the size of space of keys, this theorem means that the collision probability is neglectable for $n \ll p$, e.g., given $n = 1000$ and $p = 2^s = 2^{160}$, the collision probability is less than $\frac{2^{20}}{2^{162}} = 2^{-142}$. Note that the security of RKH is not related to the combination of the role-keys, but rely heavily on the hardness of forging $[\frac{1}{\xi + x}]G$ under the bilinear map group system.

The following theorem indicates that the role hierarchy in $RBAC$ is hidden into role-key hierarchy:

THEOREM 3. *Under the above assignment, role hierarchy is in one-to-one correspondence with key hierarchy.*

PROOF. We show that role hierarchy can be uniquely represented by key hierarchy: Assume two different roles r_i and r_j have the same key representation, i.e., $\cup_{r_i \not\preceq r_k}\{r_k\} = \cup_{r_j \not\preceq r_k}\{r_k\}$, such that $\cup_{r_i \preceq r_k}\{r_k\} = \cup_{r_j \preceq r_k}\{r_k\}$. This implies that two roles have the same seniors. But we know $r_i \in \cup_{r_i \preceq r_k}\{r_k\}$ and $r_j \in \cup_{r_j \preceq r_k}\{r_k\}$. Since $r_i \neq r_j$, we have $\cup_{r_i \preceq r_k}\{r_k\} \neq \cup_{r_j \preceq r_k}\{r_k\}$, which is a contradiction.

Conversely, we show that key hierarchy may also recover role hierarchy by the following algorithm:

1. For each role r_i, it gets the set of roles $\cup_{r_j \not\preceq r_k}\{r_k\}$ from $\sum_{r_i \not\preceq r_k} U_k$, then compute $R_i = \cup_{r_j \preceq r_k}\{r_k\} = R \setminus \cup_{r_j \not\preceq r_k}\{r_k\}$. Finally, it inserts the R_i into a record in the search table T.

2. While T is not empty, it does the following steps:

   (a) It finds the records R_i's, which include only one element, as the set of current roles C, then deletes these records from the table T;

   (b) For each record $R_i \in T$, if $R_i \setminus C = \{r_i\}$, then it outputs $r_i \prec_d r_j$ for all $r_j \in R_i \cap C$, else it erases the elements in C, i.e., $R_i = R_i \setminus C$;

Let h is the height of role hierarchy. The algorithm recurs h times and the outputs of recurrence are all edges of one layer in role hierarchy. This means that this algorithm can recover the original role hierarchy in polynomial-time. Therefore, the theorem holds. $\square$

$$
\begin{aligned}
e(W_i, B_{i,j}) \cdot e(A_{i,j}, H) &= e\left(\left[\tau_0 + \sum_{r_l \not\preceq r_i} \tau_l\right] G, \left[\frac{1}{\tau_0 + \sum\limits_{r_l \not\preceq r_i} \tau_l + x_{i,j}}\right] H\right) \cdot e\left(\left[\frac{x_{i,j}}{\tau_0 + \sum\limits_{r_l \not\preceq r_i} \tau_l + x_{i,j}}\right] G, H\right) \\
&= e(G, H)
\end{aligned} \tag{9}
$$

$$
\begin{aligned}
R' &= \left(\frac{e(W_i, C_2) \cdot e(C_1, H)}{e(G, H)}\right)^{-c} \cdot e(W_i, H)^s \\
&= \left(\frac{e(W_i, B_{i,j} + [\beta]H) \cdot e(A_{i,j} + [\alpha]W_i, H)}{e(G, H)}\right)^{-c} \cdot e(W_i, H)^s \\
&= \left(\frac{e(W_i, B_{i,j}) \cdot e(W_i, [\beta]H) \cdot e(A_{i,j}, H) \cdot e([\alpha]W_i, H)}{e(G, H)}\right)^{-c} \cdot e(W_i, H)^s \\
&= (e(W_i, [\beta]H) \cdot e([\alpha]W_i, H))^{-c} \cdot e(W_i, H)^s \\
&= e(W_i, H)^{-c(\alpha+\beta)} \cdot e(W_i, H)^s = e(W_i, H)^r.
\end{aligned} \tag{10}
$$

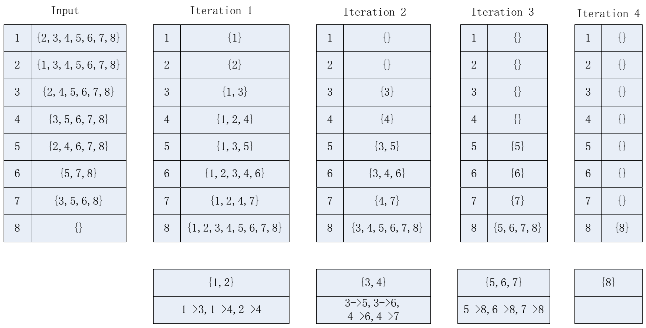

Figure 6: Example of extracting role hierarchy from key hierarchy.

Based on the algorithm in Theorem 3, Figure 6 describes an example of extracting role hierarchy from key hierarchy in Figure 5. It shows that the algorithm is effective.

We show that our scheme is secure against collusion, where two or more users, belonging to different roles, cooperate to discover a user key to which they are not entitled. To discuss the security against collusion, we make use of a hard problem, which is called Strong Diffie-Hellman (SDH) problem:

DEFINITION 6. [k-SDH problem]*: Given* $\langle G, [x]G, [x^2]G, \cdots, [x^k]G\rangle$ *to compute* $\langle c, \left[\frac{1}{x+c}\right] G\rangle$ *where* $c \in \mathbb{Z}_p^*$ *and* G *be a generator chosen from* $\mathbb{G}_1$ *(or* $\mathbb{G}_2$*).*

The standard collusion security is based on static colluders. Since we consider dynamic user management, we extend the security definition to the one that is a bit more general than in [7]. More specifically, we allow the adversary to see the role key before choosing the corrupted users. Based on this definition, we have the following theorem:

THEOREM 4. *Under the above assignment, the role-based cryptosystem (RBC) scheme is fully collusion secure. Given a role-key hierarchy* $\mathcal{H} = \{U, K, R, \preceq\}$ *with* $|U| = n$ *and* $|R| = m$*, it is* (n, n, m)*-collusion secure against framing user attack and role's privilege attack under Strong Diffie-Hellman (SDH) assumption.*

We present a proof of this theorem in Appendix B, where the whole proof for framing attack is given and the proof for role's privilege attack is stated briefly because it can be obtained from the former. The proof of this theorem indicates that the security is held even if G makes public. Moreover, this theorem clarifies that the security of this scheme is independent of the number of colluders, k. That is, the scheme can support the infinity users (at most 2^s users, where s is the security parameter) in theory. When $k = 1$, the proof of this theorem indicates that the security of the scheme against passive adversary (without collusion) is based on the hard problem $(G, [\xi]G) \rightarrow (c, [\frac{1}{\xi+c}]G)$ for $c \in_R \mathbb{Z}_p^*$.

5.3 Role-based Signature/Authencation Scheme

The above construction can be applied to derive a role-based authentication (RBA) and signature (RBS) scheme. We propose a lightweigh signature scheme to realize the anonymity and traceability. Further, this scheme can easily turn into a zero-knowledge RBA scheme. Given $\mathcal{H} = \{U, K, R, \preceq\}$, a user carries out the following process to sign a message M:

- $Sign(pk_i, sk_{i,j}, M)$: The signing algorithm takes a group public key $pk_i = (H, W_i, \{U_k\}_{r_k \in \uparrow r_i})$, a user private key $sk_{i,j} = (lab_{i,j}, A_{i,j}, B_{i,j})$, and a message $M \in \{0,1\}^*$, and proceeds as follows:

  1. Picks a random nonce $\alpha, \beta \leftarrow \mathbb{Z}_p^*$ and computes
  $$\begin{cases} C_1 = A_{i,j} + [\alpha]W_i \\ C_2 = B_{i,j} + [\beta]H \\ T = [\beta]W_i \end{cases}; \tag{11}$$
  2. Picks blinding values $r \leftarrow \mathbb{Z}_p^*$ and compute helper values S:
  $$S = e(W_i, H)^r; \tag{12}$$
  3. Computes a challenge value $c \in \mathbb{Z}_p^*$ using $Hash$:
  $$c = Hash(pk, M, C_1, C_2, T, S); \tag{13}$$
  4. Computes $s = r + c(\alpha + \beta)$:

  Finally, the signature is $\sigma \leftarrow (C_1, C_2, T, c, s)$.

- $Verify(pk_i, \sigma, M)$: The verification algorithm takes a role key pk_i, a purported signature $\sigma = (C_1, C_2, T, c, s)$, and a message $M \in \{0,1\}^*$, and proceeds the following steps:

  1. Re-derive S as:
  $$S' = \left(\frac{e(W_i, C_2) \cdot e(C_1, H)}{V}\right)^c \cdot e(W_i, H)^s; \quad (14)$$
  2. Computes $c' = Hash(pk, M, C_1, C_2, T, S')$;
  3. Check that the challenge c is correct if and only if $c = c'$. If matched, accept and reject otherwise.

  The correctness of the verification procedure is guaranteed by using Equation (9) and (10).

- $Trace(sk_{i,j}, \sigma)$: The tracing algorithm takes a set of suspicious users $RL = \{(lab_{i,j}, B_{i,j})\}$. For each element $(lab_{i,j}, B_{i,j}) \in RL$, it checks whether $B_{i,j}$ is encoded in (T, C_2) by evaluating if
$$e(W_i, C_2 - B_{i,j}) = e(T, H). \quad (15)$$
  The algorithm outputs are valid if this verification is accepted.

The security of the system is guaranteed by the fact that $A_{i,j}$ and $B_{i,j}$ are kept private. The different witnesses are generated to ensure the unlinkability as a result of the random parameters α and β. The W_i in Equation (14) and (15) determines that the role of a user cannot be modified.

The tracing algorithm may be used to realize the check of revoked users: Given a set of revoked users RL, it first verifies that the signature σ is valid by using $Verify$ algorithm; then it ensures that σ is not generated by a revoked user in terms of $Trace$ algorithm. It accepts only if both conditions are held. We reiterate the user's secret key $sk_{i,j}$ is also secure after $(lab_{i,j}, B_{i,j}) \in RL$. The reason is that the tracing or revocation check is based on $B_{i,j}$, as $A_{i,j}$ still keeps secret. Hence, the adversary cannot obtain the user's secret keys from a set of suspicious users RL.

5.4 Role-based Encryption Scheme

We can also adopt the RBC framework to build a lightweight role-based encryption (RBE) scheme, as follows:

- $Encrypt(pk_i, M)$: To encrypt the message $M \in \{0,1\}^*$, given any $pk_i = \langle H, V, W_i, \{U_k\}_{\nabla_k \in \perp r_i}\rangle$ and an empty set of revoked users $\mathcal{R} = \emptyset$, the algorithm randomly picks $t \in \mathbb{Z}_p^*$ and then computes
$$\begin{cases} C_1 &=& [t]\,W_i & \in \mathbb{G}_1 \\ C_2 &=& [t]\,H & \in \mathbb{G}_2 \\ C_3 &=& M \cdot V^t & \in \mathbb{G}_T \\ U'_k &=& [t]\,U_k \in \mathbb{G}_1 & \exists r_k \in \uparrow r_i \end{cases}. \quad (16)$$
  Finally, it outputs $\mathcal{C}_i = \langle C_1, C_2, C_3, \{U'_k\}_{r_k \in \uparrow r_i}\rangle$.

- $Decrypt(sk_{j,k}, \mathcal{C}_i)$: Given a ciphertext $\mathcal{C}_i$ from the role r_i, the k-th user in the role r_j can utilize the following equation to recover M from C_i with $dk_{j,k} = \langle A_{j,k}, B_{j,k}\rangle$, where $r_i \preceq r_j$:
$$V^t_{i,j} = e\left(C_1 + \sum_{r_l \in \Gamma(r_j, r_i)} U'_l, B_{j,k}\right) \cdot e(A_{j,k}, C_2), \quad (17)$$

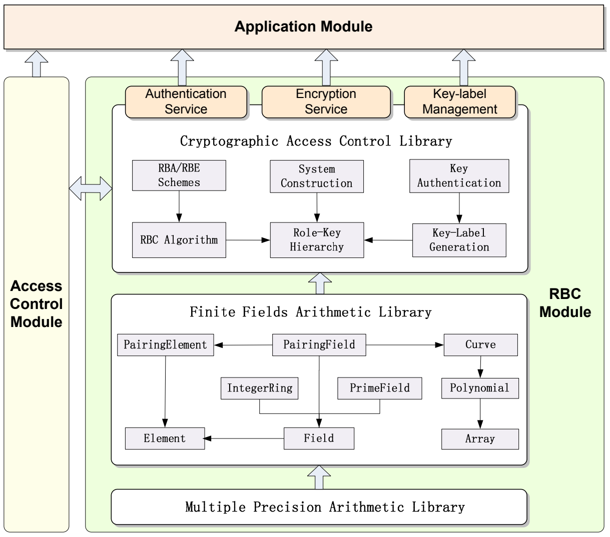

Figure 7: Cryptographic access control system based on role-key hierarchy.

  where $\Gamma(r_j, r_i)$ denotes $\cup_{r_i \preceq r_k \prec r_j}\{r_k\}$, and we have
$$\begin{aligned} \Gamma(r_j, r_i) &= \cup_{r_j \not\preceq r_k}\{r_k\} \setminus \cup_{r_i \not\preceq r_k}\{r_k\} \\ &= \cup_{r_i \preceq r_k}\{r_k\} \setminus \cup_{r_j \preceq r_k}\{r_k\} \\ &= \cup_{r_i \preceq r_k \prec r_j}\{r_k\} \end{aligned} \quad (18)$$
  in terms of Theorem 1. The algorithm outputs the session key $M = C_3 / V^t_{i,j}$.

The validity of this algorithm is guaranteed by Equation (19). Given a fixed role-key hierarchy, this algorithm achieves the constant length of ciphertexts and the optimal length of the user's secret keys, where the hidden constant relates to a couple of elements of a pairing-friendly group.

6. PERFORMANCE EVALUATION

An experimental role-based cryptosystem was implemented to test the feasibility of our schemes. This system was developed with a standard C++ language in QT environment, which supports cross-platform deployment. As shown in Figure 7, this system consists of three modules: RBC module, access control module and application module. In RBC module, we adopted GNU multiple precision arithmetic library (GMP) to handle integers of arbitrary precision. Then, a finite fields arithmetic library was constructed to realize the run-time environment of elliptic curve and pairing-based cryptosystems. In addition, a cryptographic access control library was developed based on the finite fields arithmetic library to realize various proposed RBC algorithms. Finally, the RBE/RBA/RBS algorithms worked with a lightweight access control module to provide encryption, authentication and key-label management services for the application module.

Scalability. The experimental results show our constructions are able to provide better scalability, which is an important requirement for RBAC [13]. The notion of scalability is multi-dimensional. In our schemes we can achieve scalability with respect to the number of roles, the size of role hierarchy, cardinality on user-role assignments, and so

$$
\begin{aligned}
V_{i,j}^{t} &= e\left(C_1 + \sum_{r_l \in \Gamma(r_j, r_i)} U'_l, B_{j,k}\right) \cdot e\left(A_{j,k}, C_2\right) \\
&= e\left(\left[\left(\tau_0 + \sum_{r_l \npreceq r_i} \tau_l\right) t\right] G, \left[\frac{1}{\tau_0 + \sum_{r_l \npreceq r_i} \tau_l + x_{i,j}}\right] H\right) \cdot e\left(\left[\frac{x_{i,j}}{\tau_0 + \sum_{r_l \nprec r_i} \tau_l + x_{i,j}}\right] G, [t]H\right) \\
&= e(G,H)^{\frac{(\tau_0 + \sum_{r_l \npreceq r_j} \tau_l) \cdot t}{\tau_0 + \sum_{r_l \npreceq r_j} \tau_l + x_{j,k}}} \cdot e(G,H)^{\frac{t \cdot x_{i,j}}{\tau_0 + \sum_{r_l \npreceq r_j} \tau_l + x_{j,k}}} = e(G,H)^{t}. \qquad (19)
\end{aligned}
$$

on. Moreover, our constructions support a large-size of role hierarchy with arbitrary structures. Therefore, we believe our schemes can be applied to large-scale role-based cryptosystems, such as healthcare and financial systems.

Table 1: Parameters choosing under different scales.

Parameters	Small size	Medium size	Large size
number of roles	10's	100's	1000's
number of users	10-50	50-100	100-200
height of hierarchy	1-4	5-8	9-12
total number of users	100-1,000	1,000-10,000	10,000-100,000

We can consider several degrees to measure the scalability of our method as follows: 1) small scale (10's), 2) medium scale (100's), and 3) large scale (1000's). We estimate different parameters under different scales shown in Table 1, in which we assume that the size of relations is proportional to the size of roles.

Computation Cost. The basic operation of our schemes is the computation of a multiple elliptic point in elliptic curve, namely, $[k]P$, where k is a positive integer and P is an elliptic curve point. We neglect the computation costs of an addition of elliptic points and simple modular arithmetic operations because they run fast enough. Another important operation is the computation of a bilinear map $e(\cdot,\cdot)$ between two elliptic points. Then, we use the costs of multiple operation and bilinear map operation to measure the computation complexity of our schemes.

Table 2: Comparison of computation costs on RBC. (The number of roles is m in RKH.)

	RBE	RBS/RBA
Setup	$(m+1)/1$	$(m+1)/1$
GetRkey	0/0	0/0
AddUser	2/0	2/0
Encrypt/Sign	$(m+3)/0$	4/1
Decrypt/Verify	1/2	5/3
/Trace	/	1/2

In Table 2, the costs of various algorithms in RBC, RBE, and RBS/RBA schemes are listed, where the value n/m denotes the number of multiple operations n and the number of bilinear map operations m, respectively. It is quite clear that all schemes have the low computational costs.

Communication Overhead. With the same assumption of scalability, we estimate the influence of communication overloads under the different scales. Suppose the security parameter s is 80-bits, we need the elliptic curve domain parameters over $\mathbb{F}_q$ with $|q| = 160$-bits [3]. This means that the length of integer is $l_0 = 2s$ in $\mathbb{Z}_p$. Similarly, we have $l_1 = 4s$ in $\mathbb{G}_1$, $l_2 = 20s$ in $\mathbb{G}_2$, and $l_T = 10s$ in $\mathbb{G}_T$ [4].

For RBS/RBA scheme, the communication overloads of $Sign/Interact$ is $2l_0 + 2l_1 + l_2 = 32s = 320$ bytes. For RBE scheme, the length of ciphertext is $ml_1 + l_2 + l_T = 4ms + 30s = 300 + 40m$ bytes. In terms of Table 1, we can easily compute that the overheads are increased from 0.7 KBytes(10 roles) to 40 KBytes(1000 roles).

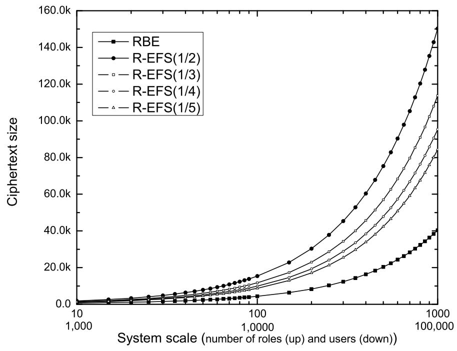

Figure 8: The ciphertext size under different scales (The size of roles changes from 10 to 1000. The number of revoked users is equal to 1/2, 1/3, 1/4, and 1/5 the number of roles.)

Figure 8 shows the change rate of ciphertext size for the RBE scheme and the R-EFS scheme. Moreover, the revocation mechanism is considered in R-EFS as well. Figure 8 indicates that the size of revoked users has more impact than other factors. Our results also indicate that the encryption based on RBE scheme performs far better than the conventional encryption file systems (EFS) with the following parameters:

- Even if we deal with a large-scale organization of 500,000 users the header of a file only requires 256 KBytes in theory (using a standard 10-bytes (80-bits) security parameter); and

[3]Elliptic curve domain parameters over $\mathbb{F}_p$ with $\lceil \log_2 p \rceil = 2t$ supply approximately t bits of security, which means that solving the logarithm problem on associated elliptic curve is believed to take approximately 2^t operations.

[4]Let the embedding degree be 5.

- The EFS with our scheme can revoke an approximate 1,000 users (some intermediate data are saved to decrypt a file faster) or 10,000 users in a compressed form at once.

7. CONCLUSION

We have proposed a role-key hierarchy structure along with hierarchical RBAC model to accommodate the requirements of cryptographic access control for large-scale systems. Based on this hierarchy model, we further proposed several practical role-based security mechanisms to support signature, authentication and encryption constructions on elliptic curve cryptosystem. Our experiments clearly demonstrated the proposed schemes are flexible and efficient enough to support large-scale systems. For our further work, we plan to accommodate other access control features of RBAC such as session management and constraints. Also, our promising results lead us to investigate how emerging distributed computing technologies such as service computing, cloud computing and mobile computing can leverage the proposed schemes with possible extensions.

8. REFERENCES

[1] S. Akl and P. Taylor. Cryptographic solution to a multilevel security problem. In *Advances in Cryptology (CRYPTO'82)*, 1982.

[2] S. Akl and P. Taylor. Cryptographic solution to a problem of access control in a hierarchy. *ACM Transaction Computer System*, 1(3):239–248, 1983.

[3] E. Bertino, N. Shang, and S. Wagstaff. An efficient time-bound hierarchical key management scheme for secure broadcasting. *IEEE Trans. on Dependable and Secure Computing*, 5(2):65–70, 2008.

[4] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *Advances in Cryptology (CRYPTO'01)*, volume 2139 of LNCS, pages 213–229, 2001.

[5] D. Boneh and M. Hamburg. Generalized identity based and broadcast encryption schemes. In *ASIACRYPT*, pages 455–470, 2008.

[6] D. Boneh and H. Shacham. Group signatures with verifier-local revocation. In *ACM Conference on Computer and Communications Security*, pages 168–177, 2004.

[7] B. W. D. Boneh, C. Gentry. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *Advances in Cryptology (CRYPTO'2005)*, volume 3621 of LNCS, pages 258–275, 2005.

[8] C. Gentry and A. Silverberg. Hierarchical id based cryptography. In *Advances in Cryptology (ASIACRYPT 2002)*, volume 2501 of LNCS, pages 548–566, 2002.

[9] E. Goh, H. Shacham, N. Modadugu, and D. Boneh. Sirius: Securing remote untrusted storage. In *Proceedings of the Internet Society (ISOC) Network and Distributed Systems Security (NDSS) Symposium*, pages 131–145, 2003.

[10] J. Jing and G.-J. Ahn. Role-based access management for ad-hoc collaborative sharing. In *Proc. of 11th Symposium on Access Control Models and Technologies (SACMAT)*, pages 200–209, 2006.

[11] X. Liang, Z. Cao, H. Lin, and J. Shao. Attribute based proxy re-encryption with delegating capabilities. In *ASIACCS*, pages 276–286, 2009.

[12] R. S. Q. Mahesh Kallahalla, Erik Riedel and K. Fu. Plutus: Scalable secure file sharing on untrusted storage. In *Proceedings of the 2nd USENIX Conference on File and Storage Technologies (FAST)*, pages 29–42, 2003.

[13] R. Sandhu, E. Coyne, H. Fenstein, and C. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.

[14] R. Sandhu, D. Ferraiolo, and D. Kuhn. The nist model for role-based access control: Towards a unified standard. In *Proceedings of 5th ACM Workshop on Role Based Access Control (RBAC'00)*, pages 47–63, 2000.

[15] W. Tzeng. A time-bound cryptographic key assignment scheme for access control in a hierarchy. *IEEE Trans. on Knowledge and Data Engineering*, 14(1):182–188, 2002.

[16] D. Wallner, E. Harder, and R. Agee. Key management for multicast: Issues and architecture. Technical Report IETF RFC 2627, In internet draft draft-waller-key-arch-01.txt, 1999.

[17] C. Wong, M. Gouda, and S. Lam. Secure group communications using key graphs. In *Proc. ACM SIGCOMM'1998*, volume 28 of ACM press, pages 68–79, 1998.

APPENDIX

A. SECURITY MODEL FOR COLLUSION

We define the security notion against collusion attacks in terms of security games between a challenger $\mathcal{B}$ and an adversary $\mathcal{A}$. We divide the users into two categories: honest users and corrupted users, so that a set of corrupted users $\mathcal{R}$ is built. Moreover, there exists many honest and corrupted users in the same role. We first define a general model against collusion attacks:

1. Initial: The challenger $\mathcal{B}$ constructs an arbitrary role hierarchy $\Psi = \langle R, \preceq \rangle$ with $|R| = m$, and then runs $Setup(s, \Psi)$ to generate the partial-order key hierarchy $\mathcal{H}$ and initial public parameters $params$, finally sends them to $\mathcal{A}$.

2. Learning: $\mathcal{A}$ adaptively issues n times queries $q_1, \cdots, q_n$ to learn the information of $\mathcal{H}$, where q_i is one of the following:

   - Honest user query ($u_{i,j} \notin \mathcal{R}$): Using $AddUser(mk, u_{i,j})$, $\mathcal{B}$ generates a new user and sends this user's label $lab_{i,j}$ to $\mathcal{A}$.
   - Corrupted user query ($u_{i,j} \in \mathcal{R}$): Using $AddUser(mk, u_{i,j})$, $\mathcal{B}$ generates a new user and returns this user's secret key $sk_{i,j}$, includes user label $lab_{i,j}$ and private key $dk_{i,j}$, to $\mathcal{A}$.

   In fact, the n users are joined into this system via n times queries. $\mathcal{A}$ ends up with a key hierarchy $\mathcal{H}$ (includes $(params, \{lab_{i,j}\}_{u_{i,j} \notin \mathcal{R}})$) and a set of colluders $\{sk_{i,j}\}_{u_{i,j} \in \mathcal{R}}$, where $|\mathcal{R}| = t$.

3. Challenge: Involves two cases:

- Framing user attack ($lab_{i,j} : u_{i,j} \notin \mathcal{R}$): $\mathcal{B}$ picks a honest user $u_{i,j}$ at random, and then sends his label $lab_{i,j}$ to $\mathcal{A}$ as the challenge.
- Role's privilege attack ($r_i \in R$): $\mathcal{B}$ picks a role $r_i \in R$ at random, and then sends the challenge r_i to $\mathcal{A}$. Note that, r_i may be either the honest roles (for avoiding the revocation) or the corrupted roles (for gain the privilege).

4. Guess: $\mathcal{A}$ outputs a guess of user key $sk_{i,j}$. $\mathcal{A}$ wins if $sk_{i,j}$ is valid, and otherwise it loses.

We denote by $Adv_{\mathcal{E},\mathcal{A}}(t,n,m)$ the advantage of adversary $\mathcal{A}$ in winning the game:

$$\begin{aligned} &Adv_{\mathcal{H},\mathcal{A}}(t,n,m) \\ = \; & \frac{1}{2} \left|\Pr[V(\mathcal{H},\mathcal{A}_{\mathcal{H}}(\mathcal{C}_i))=1] - \Pr[V(\mathcal{H},\mathcal{A}_{\mathcal{H}}(\mathcal{C}_i))=0]\right| \\ = \; & \left|\Pr[V(\mathcal{H},\mathcal{A}_{\mathcal{H}}(C_i))=1] - \frac{1}{2}\right|, \end{aligned}$$

where, $\mathcal{C}_i$ is the i-th challenge and V is a verification function of $sk_{i,j}$. We say that a construction is (t,n,m)-secure if for a security parameter s and all probabilistic polynomial time adversaries $\mathcal{A}$, $Adv_{\mathcal{E},\mathcal{A}}(t,n,m)$ is a negligible function of s.

B. SECURITY PROOF AGAINST COLLUSION ATTACKS

PROOF. Firstly, we prove the security of framing attack. Without loss of generality, we prove the security in $\mathbb{G}_1$ rather than in $\mathbb{G}_1 \times \mathbb{G}_2$. It is obvious that the latter is hard-er than the former. The security of frame attack is changed into the problem: given the number of colluders t and

$$\{W_i\}_{r_i \in R}, \{\langle x_{i_l,j_l}, A_{i_l,j_l}\rangle\}_{u_{i_l,j_l} \in \mathcal{R}},$$

it is infeasible to forge a new key $\langle x'_{i,j}, A'_{i,j}\rangle$, where $|\mathcal{R}| \le t$, and $\{x'_{i_l,j_l}\}_{u_{i_l,j_l} \in \mathcal{R}}$.

The proof is used by the reduction to absurdity. Suppose an adversary $\mathcal{A}$ can solve the above problem with the advantage ϵ, i.e., $Adv_{\mathcal{H},\mathcal{A}}(t,n,m) > \epsilon$. Using $\mathcal{A}$, we build an algorithm $\mathcal{B}$ to solves the k-SDH problem in $\mathbb{G}_1$.

Assume the algorithm $\mathcal{B}$ is input a random sequence

$$\langle G, [\xi]G, [\xi^2]G, \cdots, [\xi^k]G\rangle$$

and expect to output $\langle c, [\frac{1}{\xi+c}]G\rangle$, where ξ is unknown and $c \in \mathbb{Z}_p^*$. $\mathcal{B}$ works by interacting with $\mathcal{A}$ as follows:

1. Initial: Firstly, $\mathcal{B}$ chooses an arbitrary role hierarchy $\langle R, \preceq\rangle$ and assigns a random $\tau_i \in \mathbb{Z}_p^*$ for each role $r_i \in R$, as well as a random $\tau_0 \in \mathbb{Z}_p^*$. Let $\zeta_i = \tau_0 + \sum_{r_i \not\preceq r_k} \tau_k$ in terms of the proposed scheme. $\mathcal{B}$ selects at most $k-1$ roles as the set of corrupted roles $\{r_{c_i}\}$, and use them to a polynomial $f(x) = \prod_{i=1}^{k-1}(\zeta_{c_i} x + x_i) = \sum_{j=0}^{k-1} a_j \cdot x^j$ with $k-1$ degrees, where $(x_1, x_2, \cdots, x_{k-1})$ is a sequence chosen randomly by $\mathcal{B}$ in $\mathbb{Z}_p^*$. $\mathcal{B}$ defines
$$\begin{aligned} \overline{G} &= [f(\xi)]G = [\sum_{i=0}^{k-1} a_i \cdot \xi^i]G = \sum_{i=0}^{k-1} a_i \cdot [\xi^i]G, \\ U_i &= [\tau_i \cdot \xi]\overline{G} = [\sum_{j=0}^{k-1} \tau_i a_j \cdot \xi^{j+1}]G = \sum_{j=0}^{k-1} \tau_i a_j \cdot [\xi^{j+1}]G. \end{aligned}$$
Finally, $\mathcal{B}$ sends all U_i to $\mathcal{A}$ in $r_i \in R$.
2. Learning: involves two kinds of query:
   - Honest user query ($u_{i,j} \notin \mathcal{R}$): $\mathcal{B}$ picks a random t_i as the user label, saves it for the label set T, and returns it to $\mathcal{A}$.
   - Corrupted user query ($u_{i,j} \in \mathcal{R}$): If r_i is the corrupted roles $\{r_{c_i}\}$ and $f(x)$ has a unassigned item $(\zeta_{c_i} x + x_i)$, then $\mathcal{B}$ defines $lab_{i,j} = x_{i,j} = x_i$ and
   $$A_{i,j} = \left[\frac{x_{i,j}}{\zeta_{c_i}\xi + x_{i,j}}\right]\overline{G} = \left[\frac{x_i \cdot f(\xi)}{\zeta_{c_i}\xi + x_i}\right] G. \quad (20)$$
   $\mathcal{B}$ can compute it due to $f(x)/(\zeta_{c_i} x + x_i)$ is a computational polynomial with $k-2$ degree. $\mathcal{B}$ sends it to $\mathcal{A}$.

   Finally, after n times queries, $\mathcal{A}$ gets at most $k-1$ secret keys and a label set $T = \{t_i\}$.
3. Challenge: $\mathcal{B}$ picks a random t_s from $T = \{t_i\}$, and then sends it to $\mathcal{A}$ as the challenge label. Then $\mathcal{A}$ runs the algorithm to output a new forged key $\langle lab_{l,s}, A_{l,s}\rangle = \langle t_s, [\frac{t_s}{\zeta_l\xi + t_s}]\overline{G}\rangle$ in the role r_l.
4. Guess: $\mathcal{B}$ changes $\langle t_s, [\frac{t_s}{\zeta_l\xi+t_s}]\overline{G}\rangle$ into the representation of the generator G, $\langle t'_s, [\frac{1}{\xi+t'_s}]G\rangle$, as follows:
   $$\begin{aligned} \left[\frac{t_s}{\zeta_l\xi + t_s}\right]\overline{G} &= [t_s]\left[\frac{f(\xi)}{\zeta_l\xi + t_s}\right]G \quad (21) \\ &= [t_s]\left[\sum_{i=0}^{k-2} a'_i \cdot \xi^i + \frac{r}{\zeta_l\xi + t_s}\right]G, \end{aligned}$$
   where, we can compute $a'_0, \cdots, a'_{k-2}$ by $f(x) = \sum_{i=0}^{k-1} a_i \cdot x^i = (\sum_{i=0}^{k-2} a'_i \cdot x^i)(\zeta_l x + t_s) + r$. Such that if $t_s \ne x_l$, then $r \ne 0$. Thus, $\mathcal{B}$ can compute
   $$\begin{aligned} \left[\frac{1}{\xi + t'_s}\right] G &= [\frac{\zeta_l}{r}]\left([t_s^{-1}]A_{l,s} - \sum_{i=0}^{k-2} a'_i \cdot \left([\xi^i]G\right)\right) \\ &= [\frac{\zeta_l}{r}]\left([t_s^{-1}]\left[\frac{t_s}{\zeta_i\xi + t_s}\right]\overline{G} - \sum_{i=0}^{k-2} a'_i \cdot \left([\xi^i]G\right)\right). \end{aligned}$$
   and $t'_s = t_s/\zeta_l$. Finally, $\mathcal{B}$ outputs $\langle t'_s, [\frac{1}{\xi+t'_s}]G\rangle$.

Hence, if the adversary $\mathcal{A}$ can break RHC scheme at polynomial time in non-negligible advantage ε, then the algorithm $\mathcal{B}$ can solve k-SDH problem at the polynomial-time, that is, the advantage of algorithm $\mathcal{B}$ is $Adv_{\mathcal{H},\mathcal{A}}(k-1,n,m) > \varepsilon$. This denotes the algorithm $\mathcal{B}$ can compute k-SDH with a non-negligible success probability, which would contradict assumption.

Similarly, we also prove collusion security against the role's privilege attack, only if we change the challenge into an assigned role r_l and the adversary $\mathcal{A}$ returns a forged key $\langle t_s, [\frac{t_s}{\zeta_l\xi+t_s}]\overline{G}\rangle$ in the role r_l, as well as the other parts of this proof remain unchanged. □

It is easy to find that the security of this scheme is independent of the number of colluders under k-SDH assumption. This scheme is also secure even if the number of colluders is equal to the total number of users under $(n+1)$-SDH assumption, where $k-1=n$ and $|U|=n$.