RIKE: Using Revocable Identities to Support Key Escrow in PKIs

Nan Zhang[1,2], Jingqiang Lin[1,⋆], Jiwu Jing[1], and Neng Gao[1]

[1] State Key Lab of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100195, China
{zhangnan,linjq,jing,gaoneng}@lois.cn
[2] Graduate University of Chinese Academy of Sciences, Beijing 100049, China

Abstract. Public key infrastructures (PKIs) are proposed to provide various security services. Some security services such as confidentiality, require key escrow in certain scenarios; while some others such as non-repudiation, prohibit key escrow. Moreover, these two conflicting requirements can coexist for one user. The common solution in which each user has two certificates and an escrow authority backups all escrowed private keys for users, faces the problems of efficiency and scalability. In this paper, a novel key management infrastructure called *RIKE* is proposed to integrate the *inherent key escrow* of identity-based encryption (IBE) into PKIs. In RIKE, a user's PKI certificate also serves as a *revocable identity* to derive the user's IBE public key, and the revocation of its IBE key pair is achieved by the certificate revocation of PKIs. Therefore, the certificate binds the user with two key pairs, one of which is escrowed and the other is not. RIKE is an effective certificate-based solution and highly compatible with traditional PKIs.

Keywords: Certificate, identity-based encryption, key escrow, key management, public key infrastructure, revocation.

1 Introduction

Public key infrastructures (PKIs) are proposed to publish public keys. In PKIs, the public key of a user[1] is bound to its identity in a certificate, signed by a certification authority (CA). After querying and validating a certificate, everybody can use the contained public key for authentication, confidentiality, data integrity and non-repudiation.

Key escrow is required or prohibited in different security services, even for one user. On one hand, if a key pair is used for data encryption and decryption, key escrow is usually needed. A typical example is that, in a corporation, the

⋆ Corresponding author: The authors were partially supported by National Natural Science Foundation of China grant 70890084/G021102, 61003273 and 61003274, and Strategy Pilot Project of Chinese Academy of Sciences sub-project XDA06010702.

[1] In this paper, "user" may refer to a person, a device, an application process or any equivalent entity.

F. Bao, P. Samarati, and J. Zhou (Eds.): ACNS 2012, LNCS 7341, pp. 48–65, 2012.

backups of all employees' private keys are stored in a trusted party, called the escrow authority (EA) or the key management center (KMC); so the corporation can decrypt all encrypted data in case the employee's private key is unavailable. On the other hand, if the key pair is used to sign and verify messages for non-repudiation, key escrow is usually prohibited. For example, key escrow is forbidden or unrecommended in the digital signatures laws and the guidelines of several countries and organizations [3,11,16,17].

To satisfy the conflicting requirements of key escrow, a common solution is to let each user hold two key pairs and accordingly two PKI certificates. The key pair and the certificate used for non-repudiation, don't support key escrow; while the others do. An instruction (called the key usage extension) is embedded in each certificate to indicate the purpose of the key pair [13]. In this two-certificate solution, every user generates its own key pair for non-repudiation and the EA generates the user's key pair for other purposes. The two public keys are contained separately in two certificates. As a result, more resources are needed to sign, publish and revoke certificates. Moreover, the EA faces the problem of scalability: as new key pairs are generated to replace expired certificates and new users join the system, the task of maintaining the great amount of escrowed private keys becomes very heavy.

Identity-based encryption (IBE) is a special type of public key algorithms, with the feature of inherent key escrow. In IBE, a trusted private key generator (PKG) initializes a secret master key and publishes the corresponding pubic parameters. A user's public key can be calculated from its identity (and the pubic parameters) by other users. Thus, certificates are not needed. When receiving messages encrypted by its public key (or its identity), the user asks the PKG to generate the private key corresponding to its identity. As for key recovery, the PKG only needs to hold and protect the secret master key, to regenerate private keys for all identities (or users). However, key revocation (when a private key is compromised) is a problem in IBE, since the public key are bound to identities automatically and users are not willing to change their identities. One the contrary, there are various certificate revocation mechanisms in PKIs, applicable to different scenarios.

The above observation that IBE and PKIs have complementary advantages gives us the motivation to integrate IBE and PKIs. In this paper, we propose RIKE, using Revocable Identities of IBE to support Key Escrow in PKIs. RIKE doesn't invent any new public key algorithm; instead, it is an innovative key management infrastructure assembling the advantages of both these two cryptosystems. Each RIKE user has one certificate and the public key in it is only used for the security services prohibiting (or not requiring) key escrow. The other security services requiring key escrow, are achieved through IBE; however, the IBE public key is derived from the user's certificate, not from its real identity. Compared with traditional PKIs, RIKE provides security services with the conflicting requirements of key escrow, while each user has only one certificate. Moreover, supporting key escrow in RIKE is much easier than that in PKIs, due to the inherent key escrow of IBE.

RIKE also solves the key revocation problem of IBE, by utilizing the certificate revocation mechanisms of PKIs. Unlike deriving the public key from the receiver's unchangeable identity in IBE, the RIKE user derives the IBE public key from the receiver's certificate after validating that certificate (including checking its revocation status). If the receiver's IBE private key is compromised, the CA will revoke its certificate and then the corresponding IBE public key shall not be used. If a new certificate is issued, a new IBE key pair is available automatically while the receiver's identity keeps constant. Therefore, in RIKE, a certificate works as a "revocable identity".

Another advantage of RIKE is the high compatibility with PKIs. RIKE is implemented on the prevailing X.509-based PKIs, with new-designed certificate extensions (see Section 4). So the existing PKIs can be smoothly transferred to RIKE. If some users receive a certificate contained such extensions and do not understand the extensions or support IBE, they just ignore these extensions and process it as a common X.509 PKI certificate.

This paper is organized as follows. Section 2 surveys the related work. Section 3 presents the detailed architecture and the analysis of RIKE. Section 4 discusses how to implement RIKE using X.509 PKI certificates. Finally, Section 5 draws the conclusion.

2 Related Work

How to perfectly support key escrow in PKIs is still an open problem. The common and prevailing approach is to store the backup of a user's private key in a centralized component [15,32]. This solution is compatible with traditional PKIs, but lacks of scalability because more and more private keys are stored as the number of users increases and certificates expire. Self-escrow PKIs (SE-PKIs) are proposed [9,31] to embed the public part of trapdoor information when users generate private keys. The key recovery agent (KRA), holding the secret part of trapdoor information, can recover users' private keys. However, the specifically-designed public key algorithms are not supported by most users and then obstruct the adoption of SE-PKIs. In this work, we try to provide a solution supporting efficient key escrow and compatible with traditional PKIs.

The concept of IBE originally proposed by Shamir [33]. Boneh *et al.* [7] and Cocks [12] invented secure IBE algorithms, in which an arbitrary bit-string can be used as a public key. Hierarchical IBE [20,21] is designed to reduce the workload of the centralized PKG. All these algorithms have two basic features: (*a*) a user's identity known by others are used as the user's public key, so the certificate is eliminated; and (*b*) IBE inherently supports key escrow because all users' private keys are generated by PKGs and can be recovered by PKGs.

The inherent key escrow is usually considered as a drawback of IBE in many scenarios, because of the risk that the users' private keys may be disclosed or maliciously used by the PKG. [18,23] proposed to generate the private keys by distributed PKGs, so that the keys are still secure when the PKGs are partially compromised. Alternatively, the privilege of PKGs can be constrained. In

certificate-based encryption (CBE) [19], a user's non-escrowed secret key and a certificate from its CA are both needed to decrypt messages; and [34] designed a CBE-based proxy cryptosystem with revocable proxies. In certificateless public key cryptography (CL-PKC) [2], the private key is generated by a user and a key generating center (KGC) cooperatively. However, the feature of key escrow can be leveraged in the scenarios where key escrow is necessary, and the inherence makes the key-escrow design of RIKE concise and efficient.

The inherent binding of identities and public keys becomes a problem in IBE when the user's public key needs to be revoked (e.g., the private key is compromised), because identities are usually expected to remain constant. To deal with the revocation problem, the user's identity and a period of validity can be combined together to derive its public key [7]. Once the period expires, the key pair becomes invalid automatically; however, if the private key is compromised during the period, the key pair can not be revoked in this way. So, the period of validity shall be very short for high security, and the PKG shall generate and users shall apply for private keys very frequently. Another approach is the security mediator (SEM) architecture [5,6,26], in which an online SEM keeps a partition of each user's private key and every decryption operation requires the SEM's help. Revocation is achieved as long as the SEM stops helping the user to decrypt messages. But an always-online SEM service faces more risks, so more expensive protections are needed in practical deployment.

On the contrary, a public key (or a certificate) in PKIs is used only after its revocation status is checked. Lots of approaches are proposed to revoke PKI certificates, such as certificate revocation lists (CRLs) [13], redirect CRLs [1], the online certificate status protocol (OCSP) [29], NOVOMODO [27], certificate revocation trees (CRTs) [25] and authenticated dictionaries [30]. These revocation mechanisms have advantages in different environments, and all can be used in RIKE to revoke certificates. More detailed comparisons and evaluations of revocation mechanisms can be found in [22,28].

Some schemes are proposed to provide benefits similar to IBE, focusing on the compatibility and interoperability issues. In the RSA-based schemes [14,24], a user can encrypt messages to another user by its identity (without a certificate); or, the identity-to-key binding is implemented by online query [10]. Key escrow is not considered in these schemes. Our solution applies IBE to support key escrow in PKIs, keeping the complete compatibility with PKI certificates. For those users that do not support IBE algorithms, they can still use a certificates with the RIKE-parameter extension as a common PKI certificate.

3 RIKE: Supporting Key Escrow in PKIs

In this section, we firstly describes the background of PKIs and IBE, and the basic architecture of RIKE. Then, we extend this basic architecture to work with hierarchical PKIs and cross certification. Finally, we present the features of RIKE and the comparison with other schemes.

3.1 Background

PKIs are built based on traditional public key algorithm. A CA signs certificates, each of which binds a user's identity and its public key. Trusting the CA, a user verifies the CA's signature on a certificate and obtains another's identity and public key. Then, these certified information is used for various security services.

The notations and conceptions about PKIs are listed as follows. The superscript $\mathcal{P}$ indicates the key pairs in PKIs (to distinguish with those key pairs in IBE, for which the superscript $\mathcal{I}$ is used).

- ID_U, ID_{CA}: the identities of a user U and the CA, respectively.
- $PK^{\mathcal{P}}$: a public key of traditional public key algorithms.
- $SK^{\mathcal{P}}$: a private key of traditional public key algorithms.
- $Cert(U[,e]) = Sign_{SK^{\mathcal{P}}_{CA}}(ID_{CA}, ID_U|PK^{\mathcal{P}}[,e])$: the certificate signed by the CA, binding ID_U and $PK^{\mathcal{P}}$, with an *optional* extension e. The CA and U are called the *issuer* and the *subject* of $Cert(U[,e])$, respectively.

IBE is a special type of public key algorithms, where a public key is derived from the user's identity. A PKG firstly generates a master key and public parameters. The public parameters are publicly known, then a user can calculate another's public key based on its identity and the public parameters. Only the PKG can calculate the corresponding private key by the secret master key. The notations and conceptions about IBE are listed as follows:

- PM: the public parameters generated by the PKG.
- MK: the PKG's master key.
- $PK^{\mathcal{I}}(ID_U) = GenPK(ID_U, PM)$: the public key derived from the identity of U and PM. It can be calculated by any user.
- $SK^{\mathcal{I}}(ID_U) = GenSK(ID_U, MK)$: the private key generated by the PKG for U. Only the PKG can calculate it.

3.2 Basic RIKE

The basic idea of RIKE is to use (the hash value of) a user's PKI certificate as its identity to generate the IBE public key, used for the security services requiring key escrow. Then, each RIKE user has two key pairs but *only one* certificate. The basic architecture of RIKE is composed of a CA and an arbitrary number of users. The component responsible for signing certificates and generating escrowed private keys in RIKE, is still called the CA, to emphasizing its high compatibility with PKIs. Different from that in PKIs, the CA in RIKE owns a *PKG agent*, holding the IBE master key and generating escrowed private keys for users. Nevertheless, when RIKE is deployed, the PKG agent and the CA can be managed by two departments or implemented in one system.

We can easily implement the basic RIKE as follows, and $H(\cdot)$ is a collision-free hash function.

Initialization. The CA generates $(PK^{\mathcal{P}}_{CA}, SK^{\mathcal{P}}_{CA})$ and (PM, MK), and signs a *self-signed* certificate $Cert(CA, PM)$=$Sign_{SK^{\mathcal{P}}_{CA}}(ID_{CA}, ID_{CA}|PK^{\mathcal{P}}_{CA}, PM)$.

Then, the certificate is delivered to all users in out-of-band means, as it is done in traditional PKIs; while $SK^{\mathcal{P}}_{CA}$ and MK are known only to the CA. PM is embedded in the CA's certificate as an extension. See Section 4 for more details about the extension.

Certificate and Key Application. A user U generates $(PK^{\mathcal{P}}_U, SK^{\mathcal{P}}_U)$, and then applies $Cert(U) = Sign_{SK^{\mathcal{P}}_{CA}}(ID_{CA}, ID_U | PK^{\mathcal{P}}_{CA})$ and $SK^{\mathcal{I}}_U = SK^{\mathcal{I}}(H(Cert(U)))$ from the CA. $Cert(U)$ is published publicly by the CA, and the user keeps $SK^{\mathcal{P}}_U$ and $SK^{\mathcal{I}}_U$ secret. $(PK^{\mathcal{P}}_U, SK^{\mathcal{P}}_U)$ is called the *non-escrowed key pair* of U, and $(PK^{\mathcal{I}}_U, SK^{\mathcal{I}}_U)$ is called the *escrowed key pair* of U, where $PK^{\mathcal{I}}_U = PK^{\mathcal{I}}(H(Cert(U)))$.

Signing and Verification. U signs a message by $SK^{\mathcal{P}}_U$. Everybody can query and validate $Cert(U)$ to obtain $PK^{\mathcal{P}}_U$, and verify the signed message. Here, the validation of $Cert(U)$ includes checking the period of validity, the CA's signature on it, and its revocation status, as it does in PKIs.

Encryption and Decryption. Another user that wants to send encrypted data to U, firstly queries and validates $Cert(U)$, and calculates $PK^{\mathcal{I}}_U$ based on $Cert(U)$ and PM extracted from the CA's certificate. Then, data are encrypted by $PK^{\mathcal{I}}_U$ and sent to U. When receiving encrypted data, U decrypts them by $SK^{\mathcal{I}}_U$.

3.3 Certificate Renewal and Revocation in Basic RIKE

Before using either the public key in a certificate or the IBE public key derived from it, a user shall check the CA's signature, the certificate's period of validity and its revocation status. All certificate revocation mechanisms in PKIs can be used in RIKE. If the certificate expires or is revoked, the two public key shall not be used any more. The two key pairs may change or not after certificate renewal and revocation.

A user's certificate can be renewed when it expires. In the new certificate, the period of validity is updated, so the escrowed key pair changes automatically. But the non-escrowed key pair contained in the certificate will be either regenerated or kept unchanged (if it is still considered as secure).

A user's certificate is revoked and a new one with valid information is usually signed, when (a) the information in the certificate becomes invalid (e.g., its affiliation changes), or (b) the non-escrowed key pair or the escrowed key pair is (suspected to be) compromised.

- If a certificate is revoked due to the invalid information, the user's non-escrowed key pair may keep unchanged in the new certificate, but the escrowed key pair derived from the new certificate becomes different.
- If the certificate is revoked due to the compromise of the non-escrowed key pair, both the two key pairs will change even if the escrowed key pair is not compromised.
- If the certificate is revoked due to the compromise of the escrowed key pair, the user's non-escrowed key pair may keep unchanged in the new certificate (but at least one bit in the certificate shall be modified, e.g., the period of validity, so the escrowed key pair will change).

In summary, the non-escrowed key pair does not change unless it needs to change, while the escrowed key pair always changes once the certificate is replaced by a new one. But the sender who uses another user's IBE public key doesn't need to know whether the key pair has changed or not, it only validates the recipient's certificate (just as what they do in traditional PKIs) and derives the IBE public key from it. There is no extra burden for RIKE users to deal with certificate renewal and revocation.

3.4 Hierarchical RIKE

PKIs are usually built hierarchically [13], as a user applies its certificate from another user. For example, a CA (called the *root CA* in hierarchical PKIs) generates the self-signed certificate, and signs certificates for the 2nd-level users, some of which sign certificates for other users. The user that signs certificates for others, is also called a *subordinate CA*. In this paper, we call it a user or a CA alternatively according to the context. This structure can be easily extended to support more levels, where each user applies for a certificate from an upper-level user or the root CA.

The following new notations are used in a hierarchical PKI:

- $U_{j,k}$: the k^{th} j^{th}-level user.
- $Cert(U_{j,k}) = Sign_{SK^{\mathcal{P}}_{U_{j-1,k'}}}(ID_{Isr}, ID_{U_{j,k}}|PK^{\mathcal{P}}_{U_{j,k}})$: the certificate of $U_{j,k}$ signed by Isr, which is another user $U_{j-1,k'}$ or the root CA (when $j = 2$).

In order to validate $Cert(U_{j,k})$, a vector of certificates (called the *certificate chain*) are needed: $\overrightarrow{Cert}(U_{j,k}) = (Cert(U_{2,k_2}), \cdots, Cert(U_{j-1,k_{j-1}}), Cert(U_{j,k}))$, where the issuer of each certificate is the subject of the preceding one and $Cert(U_{2,k_2})$ is signed by the root CA. Every user has been already configured with the root CA's self-signed certificate and uses it to validate $Cert(U_{2,k_2})$, and then repeatedly uses a validated certificate to validate the next one in the vector until $Cert(U_{j,k})$ is validated.

The hierarchical structure brings benefits. The root CA's workload is distributed among lower-level CAs, and users can apply for certificates locally. The root CA serves only a few users (or CAs) and then works off-line in most time to reduce attack risks. If a lower-level CA is compromised, only a limited number of users are impacted.

To build RIKE on hierarchical PKIs, we need to find a compatible way to manage IBE key pairs (or escrowed key pairs). Two intuitive and simple approaches are listed as follows:

- Only the root CA owns the PKG agent, which generates the escrowed private keys for all users. The IBE public parameters are embedded only in the root CA's self-signed certificate. Or,
- Each CA owns its PKG agent, which generates the escrowed private keys for its users only. Then, each CA's certificate contains the IBE public parameters of its own PKG agent.

However, neither of these simple approaches works well. In the first approach, the root CA takes the workload of generating escrowed private keys for all users, which violates the intentions of hierarchical PKIs. In the second one, an upper-level PKG agent cannot recover the private keys generated by the lower-level PKG agents, while sometimes centralized key escrow is needed.

We propose to build RIKE by combining hierarchical PKIs and hierarchical IBE [20,21]. Hierarchical IBE works as follows. For example, a PKG (called the *root PKG*) generates its master key and the public key parameters, and generates IBE private keys for the 2nd-level users, some of which generate IBE private keys for other users. The user that generates IBE private keys for others is called a *subordinated PKG*. In this paper, we call it a user or a PKG alternatively according to the context. The structure can be easily extended to support more levels, where each user applies its IBE private key from an upper-level user or the root PKG. Note that all users' private keys are generated by the secret master key directly or indirectly, so the root PKG can recover any user's private key.

In hierarchical IBE, each user's public key is derived from its *identity chain*, and the following new notations are introduced:

- $ID_{U_{j,k}}$: the identity of the k^{th} j^{th}-level user $U_{j,k}$; particularly, the root PKG can be denoted as $U_{1,1}$ and its identity is null.
- $\overrightarrow{ID}_{U_{j,k}} = (\overrightarrow{ID}_{\star}, ID_{U_{j,k}})$: the identity chain of $U_{j,k}$, where $\overrightarrow{ID}_{\star}$ is (a) the identity chain of another user $U_{j-1,k'}$ if $U_{j,k}$ applies its IBE private key from $U_{j-1,k'}$, or (b) null if $U_{j,k}$ applies it from the root PKG (i.e, the root PKG's identity chain is also null).

Thus, the identity chain of $ID_{U_{j,k}}$ is $(ID_{U_{2,k_2}}, \cdots, ID_{U_{j-1,k_{j-1}}}, ID_{U_{j,k}})$, and its private key is generated by $U_{j-1,k_{j-1}}$ as follows:

- $PK^{\mathcal{I}}(\overrightarrow{ID}_{U_{j,k}}) = GenPK((ID_{U_{2,k_2}}, \cdots, ID_{U_{j-1,k_{j-1}}}, ID_{U_{j,k}}), PM)$: the public key of $U_{j,k}$ derived from $\overrightarrow{ID}_{U_{j,k}}$ and PM.
- $SK^{\mathcal{I}}(\overrightarrow{ID}_{U_{j,k}}) = GenSK(ID_{U_{j,k}}, SK^{\mathcal{I}}(\overrightarrow{ID}_{U_{j-1,k_{j-1}}}))$: the private key of $U_{j,k}$ generated by $U_{j-1,k_{j-1}}$. Note that $SK^{\mathcal{I}}(\overrightarrow{ID}_{U_{1,1}}) = MK$ is the root PKG's master key.

Finally, hierarchical RIKE is built on hierarchical PKIs, where each CA owns its PKG agent and these PKG agents work as the PKGs of hierarchical IBE. Only the root PKG agent publishes the IBE public parameters in the root CA's self-signed certificate (See Section 4 for details). With the same IBE public parameters, the hash values of a user's certificate chain are used to generate its IBE public key. Here, we firstly define $H(\overrightarrow{Cert}(U_{j,k})) = (H(Cert(U_{2,k_2})), \cdots, H(Cert(U_{j-1,k_{j-1}})), H(Cert(U_{j,k})))$.

- The root CA generates $(PK^{\mathcal{P}}_{CA}, SK^{\mathcal{P}}_{CA})$ and (PM, MK), and signs a self-signed certificate $Cert(CA, PM) = Sign_{SK^{\mathcal{P}}_{CA}}(ID_{CA}, ID_{CA}|PK^{\mathcal{P}}_{CA}, PM)$.
- A 2nd-level user $U_{2,k}$ generates $(PK^{\mathcal{P}}_{U_{2,k}}, SK^{\mathcal{P}}_{U_{2,k}})$, and applies $Cert(U_{2,k}) = Sign_{SK^{\mathcal{P}}_{CA}}(ID_{CA}, ID_{U_{2,k}}|PK^{\mathcal{P}}_{U_{2,k}})$ and $SK^{\mathcal{I}}_{U_{2,k}} = SK^{\mathcal{I}}(H(\overrightarrow{Cert}(U_{2,k})))$ from

the root CA. $(PK^{\mathcal{P}}_{U_{2,k}}, SK^{\mathcal{P}}_{U_{2,k}})$ is the non-escrowed key pair of $U_{2,k}$, and $(PK^{\mathcal{I}}_{U_{2,k}}, SK^{\mathcal{I}}_{U_{2,k}})$ is the escrowed key pair of $U_{2,k}$, where $PK^{\mathcal{I}}_{U_{2,k}} = PK^{\mathcal{I}}(H(\overrightarrow{Cert}(U_{2,k})))$.

- A 3rd-level user $U_{3,k}$ generates $(PK^{\mathcal{P}}_{U_{3,k}}, SK^{\mathcal{P}}_{U_{3,k}})$, and applies $Cert(U_{3,k}) = Sign_{SK^{\mathcal{P}}_{U_{2,k'}}}(ID_{U_{2,k'}}, ID_{U_{3,k}}|PK^{\mathcal{P}}_{U_{3,k}})$ and $SK^{\mathcal{I}}_{U_{3,k}} = SK^{\mathcal{I}}(H(\overrightarrow{Cert}(U_{3,k})))$ from a 2nd-level user $U_{2,k'}$.
- Any user can follow the process above to generate its non-escrowed key pair and apply its certificate and escrowed key pair.

Hierarchical RIKE has both the features of distributed workload and centralized key escrow. Hierarchical RIKE distributes the workload among subordinate PKG agents of all levels. Each subordinate PKG agent is responsible for generating the escrowed private keys of only the users directly subordinated to it. At the same time, hierarchical RIKE gives the root PKG agent the ability to recover the escrowed key pairs of all users. Given a user's certificate chain, its IBE private key can be regenerated by the root PKG agent's master key.

3.5 Hierarchical RIKE with Cross Certification

Theoretically, one hierarchical PKI (and then hierarchical RIKE) with one root CA can serve all users in the world. However, there are lots of root CAs with different self-signed certificates in the real world. Usually, a user is configured with a certificate trust list (CTL), a limited set of self-signed certificates. Users apply for certificates from different root CAs (directly or indirectly) and have different CTLs. Thus, if a user U receives $\overrightarrow{Cert}(U')$ which is signed by a root CA not in the CTL of U, it cannot validate $Cert(U')$ and communicate with U' securely. Note that self-signed certificates shall be delivered in out-of-band means and be configured carefully, so a user doesn't change its CTL rashly.

Cross certification [13] helps a user validate certificates signed by a root CA not in its CTL. As shown in Figure 1, $Cert(U')$ is signed by $RootCA_2$ (indirectly), while the CTL of U contains the self-signed certificate of $RootCA_1$ only. To help U validate $Cert(U')$, $RootCA_1$ signs a *cross certificate* $CrsCert(CA') =$

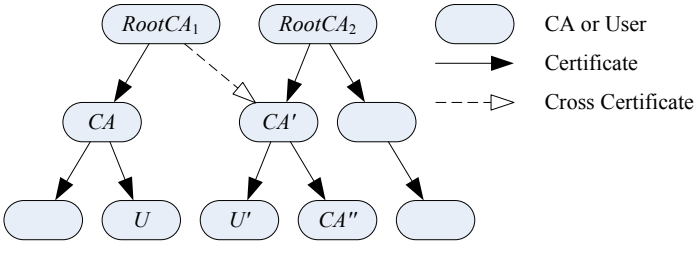

Fig. 1. Cross Certification and Cross Certificate

$Sign_{SK^{\mathcal{P}}_{RootCA_1}}(ID_{RootCA_1}, ID_{CA'}|PK^{\mathcal{P}}_{CA'})$, so U can validate $Cert(U')$ using the certificate chain $(CrsCert(CA'), Cert(U'))$.

A cross certificate is the same as a common certificate signed to a CA, except that the subject of the cross certificate is a CA which (a) already has a valid certificate and (b) has signed certificates for users. On receiving a certificate, a user cannot distinguish whether it is a cross certificate or not. Moreover, the subject (or the issuer) of a cross certificate may be a root or non-root CA, and cross certification may be bidirectional or not. For example, in Figure 1, CA' may also sign a cross certificate for $RootCA_1$ or not.

When cross certificates are issued, a user will have multiple certificate chains. For example, in Figure 1, U' has two certificate chains $(Cert(CA'), Cert(U'))$ and $(CrsCert(CA'), Cert(U'))$: one is validated by $Cert(RootCA_2)$ and the other is done by $Cert(RootCA_1)$.

The coexisting multiple certificate chains lead to two problems, if the method in Section 3.4 is directly used to derive a user's IBE public key. Firstly, different certificate chains result in different escrowed key pairs, all users (and CAs) subordinated to the subject of the cross certificate have to apply new IBE private keys after each cross certification happens. Secondly, given a user, some of its private keys are escrowed in PKI components managed by other organizations. The consequence is that the user's own organization can not recover those private keys. For example, in Figure 1, the private key that decrypts the encrypted data sent from U to U' will be escrowed only in the PKG agent of $RootCA_1$. Note that U' is a user of $RootCA_2$, but $RootCA_2$ can not recover this private key.

The solution is to always derive user's IBE public key from the same certificate chain. We choose the one in which there is no cross certificate and name it the *primary certificate chain* of a user. The user's escrowed private key is only generated based on the primary certificate chain.

To achieve this aim, the cross certificate carries the information of (a) the IBE public parameters of the root CA (or the PKG agent) in the subject's primary certificate chain and (b) the primary certificate chain of the subject (i.e., the hash values of certificates from the 2nd-level CA to the certificated subject, called the *ID-prefix* in this paper). The above information is embedded in the cross certificate[2] as an extension called the *RIKE-parameter* extension (see Section 4 for details). The verifier can use the above information to derive the same IBE public key as that from the primary certificate chain.

For example, in Figure 1, $(Cert(CA'), Cert(U'))$ is the primary certificate chain of U', and the cross certificate $CrsCert(CA')$ contains the IBE public parameters of $RootCA_2$ and the ID-prefix of CA' (i.e., $H(\overrightarrow{Cert}(CA'))$). So the IBE public key of U' is always derived from $(H(Cert(CA')), H(Cert(U')))$ and PM_{RootCA_2}, even if U validates the certificate chain $(CrsCert(CA'), Cert(U'))$ of U' by $Cert(RootCA_1)$, because $H(\overrightarrow{Cert}(CA'))$ and PM_{RootCA_2} are embedded in $CrsCert(CA')$ as a certificate extension.

[2] In RIKE, before a CA signs a cross certificate to another CA, it needs to query the primary certificate chain of the subject CA, to obtain the information.

In particular, Algorithm 1 is used to derive the IBE public key of U', when a verifier U receives a certificate chain $\overrightarrow{Cert}(U')$ different from the primary certificate chain of U'. U can reassemble (the hash values of) the primary certificate chain of U', when validating $\overrightarrow{Cert}(U')$. During this process, if $Cert_i$ is a cross certificate (containing the IBE public parameters PM and the ID-prefix), the IBE public parameters used by the verifier is substituted by PM and the ID-prefix is used to reassemble the identity chain of U'. Note that the algorithm works even when there are multiple cross certificates in $\overrightarrow{Cert}(U')$.

Algorithm 1. The Derivation of IBE Public Key

Input: The certificate chain of U', $\overrightarrow{Cert}(U') = (Cert_2, Cert_3, \cdots, Cert_j)$;
The self-signed certificate in the CTL of U, $Cert_1$;
Output: The IBE public key of U', $PK^{\mathcal{I}}_{U'}$;

$PM = PMField(RikeParamExt(Cert_1))$;
// PM is set to the IBE public parameters in $Cert_1$.
$\overrightarrow{ID} =$ null;
$IsrCert = Cert_1$;
for $(i = 2; i \leq j; i++)$ **do**
  if $Verify(IsrCert, Cert_i)$ **then**
    // $Cert_i$ is verified by $IsrCert$.
    if $e = RikeParamExt(Cert_i)$ **then**
      // $Cert_i$ contains a RIKE parameter extension e.
      $PM = PMField(e)$; // PM is set to the IBE public parameters in $Cert_i$.
      $\overrightarrow{ID} = IDPrefixField(e)$; // $\overrightarrow{ID}$ is set to the ID-prefix in $Cert_i$.
    else
      // No RIKE parameter extension in $Cert_i$.
      $\overrightarrow{ID} = AppendID(\overrightarrow{ID}, Hash(Cert_i))$;
    end if
    $IsrCert = Cert_i$; // To verify the next certificate.
  else
    return null;
  end if
end for
return $PK^{\mathcal{I}}_{U'} = GenPK(\overrightarrow{ID}, PM)$;

In Section 3.4, to guarantee the root CA (or the PKG agent) can recover all users' private keys, we required that the IBE public key parameters are only embedded in the root CA's self-signed certificate. However, with the RIKE-parameter extension in cross certificates, the IBE public key parameters may actually be obtained from a non-self-signed cross certificate. A new risk appears that some subordinate CA may maliciously embed different IBE public parameters when signing certificates for users. Note that a verifier cannot distinguish such certificates from a cross certificate. Thus, the root PKG agent can not recover the users' IBE private keys and the centralized key escrow is undermined.

Another certificate extension called the *RIKE-parameter-lock* extension, is proposed to avoid the above risk. Once this extension is set in a certificate, the subject CA and all its (directly and indirectly) subordinated CAs shall not issue a certificate with different IBE public parameters; otherwise, such a certificate is considered as invalid. Of course, those CAs are deprived of the privilege to issue cross certificates. It is reasonable, because the potentially malicious CAs should not be granted this privilege.

3.6 Certificate Renewal and Revocation in Hierarchical RIKE

In hierarchical RIKE, certificate renewal and revocation are somehow more complicated than those in basic RIKE. If the renewed (or revoked) certificate is held by a bottom-level user, the cases are the same as in basic RIKE discussed in Section 3.3.

However, if the renewed (or revoked) certificate is held by a CA, the two key pairs of the CA may change (as discussed in Section 3.3) and then impact the users subordinated to the CA. If the non-escrowed key pair of the CA (i.e., the key pair to sign and verify certificates) is changed, the CA needs to revoke all certificates it signed before. It is the same as in hierarchical PKIs, so we do not discuss this case here. The escrowed key pair of the CA may change, when the CA's certificate is renewed or revoked. Three cases are analyzed as follows:

Case 1. The renewed (or revoked) certificate is not a cross certificate, and neither the CA nor its (directly and indirectly) subordinate CAs hold cross certificates. The subordinate CAs' primary certificate chains changes because the CA's certificate changes. So all their escrowed key pairs are changed.

Case 2. The renewed (or revoked) certificate is not a cross certificate, but the CA or its (directly or indirectly) subordinate CA holds a cross certificate. In addition to all these CAs' escrowed key pairs are changed as in Case 1, the cross certificate shall be revoked and a new cross certificate shall be signed, because the ID-prefix in it needs to be updated.

Case 3. The renewed (or revoked) certificate is a cross certificate signed to the CA. Since the CA's is primary certificate chain doesn't depend on any cross certificate, its escrowed key pair is kept unchanged. So the escrowed keys of its subordinate CAs are also kept unchanged.

3.7 Features of RIKE

In RIKE, each user holds two key pairs but only one certificate: the non-escrowed key pair $(PK_U^{\mathcal{P}}, SK_U^{\mathcal{P}})$ is used in security services (e.g., non-repudiation) where key escrow is prohibited, and the escrowed key pair $(PK_U^{\mathcal{I}}, SK_U^{\mathcal{I}})$ is used in security services (e.g., confidentiality) where key escrow is required. We summarize the features of RIKE as follows.

Inherent Key Escrow. RIKE carries forward the inherent-key-escrow feature of IBE, in which all users' private keys are generated by the PKG with the

secret master key. Therefore, unlike the EA in PKIs that stores all users's private keys, the CA (or the PKG agent) in RIKE only stores the secret master key itself and avoids the problem of scalability when the user amount becomes enormous.

Effective Certificate-Based Solution. RIKE extends traditional PKIs by creatively leveraging the PKI certificates as revocable identities to support both key escrow and non-repudiation without coming into conflict. For each user, $PK_U^{\mathcal{P}}$ and $PK_U^{\mathcal{I}}$ are published together in one certificate. These two public keys' integrity and validity are guaranteed simultaneously by validating this certificate. In this way, RIKE does not bring extra communications and validations to obtain $PK_U^{\mathcal{I}}$. Therefore, RIKE is an effective solution, especially in the environments where resources are limited.

Compatibility with Traditional PKIs. RIKE is completely compatible with all policies and procedures in traditional PKIs to create, manage, distribute, use, store and revoke certificates. As a result, for the security services without key escrow, RIKE and PKIs work with thorough interoperability. The transfer from a traditional PKI to RIKE is very simple and straightforward: the root CA (or the PKG agent) generates (PM, MK) and signs a new self-signed certificate containing the essential extensions (details in Section 4), and then generates escrowed key pairs for users and subordinate CAs.

Revocable Identities. Although RIKE borrows the key generation mechanism of IBE, RIKE does not suffer from the key revocation problem as IBE does. The reason is that $PK_U^{\mathcal{I}}$ is not derived from the user's real identity, but from the user's PKI certificate. When the certificate has been revoked and replaced by a new one, $PK_U^{\mathcal{I}}$ will change automatically. Therefore, the revocation of the escrowed key pair is implemented easily by certificate revocation, for which there are already abundant approaches. In this way, RIKE supports the "revocable identity" of the user (not the real identity, but the "identity" in the perspective of IBE).

Algorithm-Independency. RIKE combines the advantages of PKIs and IBE and does not pose any additional requirements on the cryptographic algorithms. Any algorithm applicable in traditional PKIs and IBE can be used in RIKE for signature and encryption, respectively. In a word, RIKE is an algorithm-independent framework and the algorithms can be adaptively chosen in different implementations.

3.8 Comparisons with other Schemes

RIKE vs. PKI. The most straightforward way to satisfy the conflicting key escrow requirements in PKIs is to use two key pairs and two certificates in parallel. One of the two key pairs is generated and escrowed by the EA, and can be recovered by it when needed. A key usage extension is embedded in each certificate to indicate the key pair's purpose.

We compare RIKE with PKIs in three aspects as follows, showing that RIKE is more efficient than PKIs in both client-side and server-side. Firstly, to satisfy

the conflicting key escrow requirements, almost all the components in PKIs need to be scaled up. The CA shall have the ability to sign twice as many certificates as before. Resources of certificate distribution and revocation shall also increase twofold. In contrast, RIKE supports two key pairs implicitly in only one certificate. Almost no extra resource pressure is applied on PKI components and no additional certificate distribution is needed.

Secondly, in PKIs, both the two certificates of each user shall be obtained by (or transmitted to) other users. So the communication cost doubles, which impedes this solution in bandwidth-limited applications. In RIKE, no additional certificates of users are needed by encrypted-message senders.

Finally, the EA in PKIs needs to appropriately store all users' escrowed keys in a well-protected repository. In practise, besides currently valid keys, the EA also needs to store all historical keys (out-of-date keys and revoked ones), which are still useful to decrypt the old ciphertexts created when those keys were valid. As time goes on, more and more historical keys will be accumulated. Furthermore, all these private keys should be well protected with confidentiality. In RIKE, the PKG agent only stores the IBE master key in stead of all users' private keys. Whenever a specific private key shall be recovered, the PKG agent regenerates it by the IBE master key and the corresponding certificate chain. Since all the current and historical certificates are stored in plaintext, only one secret master key needs to protect, which is much simpler and more trustworthy than protecting a huge and accumulating set of private keys.

RIKE vs. SE-PKI. SE-PKIs enable PKIs to recover the private keys of all users. All the users' private keys are escrowed by the KRA which is independent from the CA. The KRA generates its own key pair (the private key SK_{KRA} and the pubic key PK_{KRA}) and provides PK_{KRA} as a parameter for users to generate their key pairs. In this way, a trapdoor is placed in the user key generation process. The user's private key can be calculated by the KRA with SK_{KRA}, so key escrow is achieved without storing and managing all users' private keys.

However, SE-PKIs also have limitations where RIKE has advantages. SE-PKIs escrow all users' private keys without differentiation, so it can not solve the conflict between key escrow and non-repudiation requirements. If SE-PKIs are adopted in traditional PKIs, each user still needs to apply for and hold two certificates. But each RIKE user only holds one certificate. Moreover, in a system with a huge amount of users, the centralized KRA in SE-PKIs undergoes a heavy workload, whereas the distributed PKGs in hierarchical RIKE (or hierarchical IBE) share the workload.

SE-PKIs depend on specially-designed algorithms, and the key generation and encryption algorithms are not compatible with the current widely-adopted PKIs'. On the contrary, RIKE is an algorithm-independent framework and could be smoothly transferred from the existing PKI systems.

Therefore, compared with SE-PKIs, RIKE is more suitable for large-scale cases and more compatible with legacy PKI systems.

RIKE vs. IBE As described above, the key idea of RIKE is the integration of PKIs and IBE by using the PKI certificate (actually, its hash value) as the "identity" in IBE. IBE provides the good feature that the sender can obtain the recipient's public key from the recipient's identity (an arbitrary bit-string, e.g. name or email address), without an online lookup. RIKE inherits this feature. So the sender obtains the recipient's escrowed public key from the recipient's PKI certificate, eliminating the additional certificate to carry it.

The major obstacle for IBE to become a fully-blown public key cryptosystem is its lack of key revocation mechanism which is necessary in practice. In IBE, the key pair is hard to revoke, because the public key is one-to-one bound with a user's identity and changing identity brings unacceptable inconvenience to the user. In contrast, RIKE supports key revocation without changing the user's identity. By leveraging the revocation mechanism of PKIs, RIKE converts PKI certificates into revocable identities.

Strictly speaking, RIKE borrows the IBE's spirit of using an arbitrary bit-string as a public key, and uses a PKI user's certificate as its identity. Email address is usually accepted as an IBE identity, because it is already commonly held and easy for human to remember and input. When RIKE is deployed for current PKI users, they have already held others' certificates and these certificates are used as IBE public keys automatically by the client applications. Therefore, although the revocable identity of RIKE is much longer than the identity of IBE, its usability is not reduced.

4 X.509-Based RIKE

In this section, we discuss how to use X.509 PKI certificates to implement RIKE, by defining two new certificate extensions, namely, the RIKE-parameter extension and the RIKE-parameter-lock extension.

The descriptions in ASN.1 syntax are as follows.

```
-- The RIKE-parameter extension
RIKEParameters ::= SEQUENCE {
    ibeAlgorithm              OBJECT IDENTIFIER,
    ibePublicParameterData    OCTET STRING,
    hashAlgorithm             OBJECT IDENTIFIER,
    idPrefix                  IDPrefix OPTIONAL }
IDPrefix ::= SEQUENCE SIZE (1..MAX) OF OCTET STRING

-- The RIKE-parameter-lock extension
RIKEParamLock ::= BOOLEAN
```

The RIKE-parameter extension `RIKEParameters` can be used in two kinds of certificates: self-signed certificates and cross certificates. The first three fields are mandatory. The field `ibeAlgorithm` and the filed `ibePublicParameterData` describe the IBE algorithm and its detailed public parameters, and the structure of `ibePublicParameterData` depends on which algorithm is used [4,8]. The

field `hashAlgorithm` specifies the hash function to convert a certificate into a revocable identity. The field `idPrefix`, which is a sequence of hash values corresponding to the certificates in the subject's primary certificate chain, only exists in cross certificates.

The RIKE-parameter-lock extension `RIKEParamLock` is a boolean value to indicate whether the RIKE parameters are locked or not. When this extension is set to true in a certificate, any other certificate following the certificate in a certificate chain, is considered invalid if it has the RIKE-parameter extension (with different RIKE parameters).

With the above extensions, a PKI can be transferred to RIKE simply and smoothly. In particular, to deploy hierarchical RIKE, the root PKG agent generates (PM, MK) and signs a new self-signed certificate with a RIKE-parameter extension. After updating this certificate in their CTLs, all the users supporting IBE algorithms can encrypt messages by the recipients' IBE public keys derived from their existing PKI certificates.

5 Conclusions

In this paper, we integrate PKIs and IBE into a novel key management scheme called RIKE, which leverages revocable identities to support key escrow in PKIs. As an innovative key management infrastructure, RIKE satisfies the conflicting requirements of key escrow, and reduces the cost of managing key pairs and certificates. Each RIKE user holds two key pairs, one of which is escrowed and the other is non-escrowed, with only one certificate. This efficient scheme assembles the advantages of both the two cryptosystems and compatibly works with hierarchical PKIs.

References

1. Adams, C., Zuccherato, R.: A general, flexible approach to certificate revocation. Technical report, Entrust (1998)
2. Al-Riyami, S.S., Paterson, K.G.: Certificateless Public Key Cryptography. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 452–473. Springer, Heidelberg (2003)
3. Asia-Pacific Economic Cooperation (APEC). Guidelines for schemes to issue certificates capable of being used in cross jurisdiction ecommerce (2004)
4. Appenzeller, G., Martin, L.: IETF RFC 5408: Identity-based encryption architecture and supporting data structures (2009)
5. Baek, J., Zheng, Y.: Identity-Based Threshold Decryption. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 262–276. Springer, Heidelberg (2004)
6. Boneh, D., Ding, X., Tsudik, G., Wong, M.: A method for fast revocation of public key certificates and security capabilities. In: 10th USENIX Security Symposium, pp. 297–308 (2001)
7. Boneh, D., Franklin, M.: Identity-Based Encryption from the Weil Pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)

8. Boyen, X., Martin, L.: IETF RFC 5091: Identity-based cryptography standard (IBCS) #1: Supersingular curve implementations of the BF and BB1 cryptosystems (2007)
9. Brown, J., Gonzalez Nieto, J., Boyd, C.: Efficient and secure self-escrowed public-key infrastructures. In: 2nd ACM Symposium on Information, Computer and Communications Security, pp. 284–294 (2007)
10. Callas, J.: Identity-based encryption with conventional public-key infrastructure. In: 4th Annual PKI Workshop, pp. 98–111 (2005)
11. China. Electronic signature law (2004)
12. Cocks, C.: An Identity Based Encryption Scheme Based on Quadratic Residues. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, pp. 360–363. Springer, Heidelberg (2001)
13. Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., Polk, W.: IETF RFC 5280: Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile (2008)
14. Ding, X., Tsudik, G.: Simple Identity-Based Cryptography with Mediated RSA. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 193–210. Springer, Heidelberg (2003)
15. Entrust. Entrust authority digital certificate solution (2012)
16. European Telecommunications Standards Institute (ETSI). Policy requirements for certification authorities issuing qualified certificates (2000)
17. European Union (EU). Directive on a community framework for electronic signatures (1999)
18. Geisler, M., Smart, N.P.: Distributing the Key Distribution Centre in Sakai–Kasahara Based Systems. In: Parker, M.G. (ed.) Cryptography and Coding 2009. LNCS, vol. 5921, pp. 252–262. Springer, Heidelberg (2009)
19. Gentry, C.: Certificate-Based Encryption and the Certificate Revocation Problem. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 272–293. Springer, Heidelberg (2003)
20. Gentry, C., Silverberg, A.: Hierarchical ID-Based Cryptography. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (2002)
21. Horwitz, J., Lynn, B.: Toward Hierarchical Identity-Based Encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 466–481. Springer, Heidelberg (2002)
22. Iliadis, J., Spinellis, D., Katsikas, S., Gritzalis, D., Preneel, B.: Evaluating certificate status information mechanisms. In: 7th ACM Conference on Computer and Communications Security, pp. 1–8 (2000)
23. Kate, A., Goldberg, I.: Distributed Private-Key Generators for Identity-Based Cryptography. In: Garay, J.A., De Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 436–453. Springer, Heidelberg (2010)
24. Khurana, H., Basney, J.: On the risks of IBE. In: International Workshop on Applied PKC, pp. 1–10 (2006)
25. Kocher, P.C.: On Certificate Revocation and Validation. In: Hirschfeld, R. (ed.) FC 1998. LNCS, vol. 1465, pp. 172–177. Springer, Heidelberg (1998)
26. Libert, B., Quisquater, J.-J.: Efficient revocation and threshold pairing based cryptosystems. In: 22nd Annual ACM Symposium on Principles of Distributed Computing, pp. 163–171 (2003)
27. Micali, S.: NOVOMODO: Scalable certificate validation and simplified PKI management. In: 1st Annual PKI Workshop, pp. 15–25 (2002)

28. Myers, M.: Revocation: Options and Challenges. In: Hirschfeld, R. (ed.) FC 1998. LNCS, vol. 1465, pp. 165–171. Springer, Heidelberg (1998)
29. Myers, M., Ankney, R., Malpani, A., Galperin, S., Adams, C.: IETF RFC 2560: X.509 Internet public key infrastructure online certificate status protocol - OCSP (1999)
30. Naor, M., Nissim, K.: Certificate revocation and certificate update. In: 7th USENIX Security Symposium, pp. 217–228 (1998)
31. Paillier, P., Yung, M.: Self-Escrowed Public-Key Infrastructures. In: Song, J.S. (ed.) ICISC 1999. LNCS, vol. 1787, pp. 257–268. Springer, Heidelberg (2000)
32. RSA, the security division of EMC. RSA digital certificate solution (2012)
33. Shamir, A.: Identity-Based Cryptosystems and Signature Schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
34. Wang, L., Shao, J., Cao, Z., Mambo, M., Yamamura, A.: A Certificate-Based Proxy Cryptosystem with Revocable Proxy Decryption Power. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 297–311. Springer, Heidelberg (2007)