Identity-Embedding Method for Decentralized Public-Key Infrastructure

Anada, Hiroaki
Institute of Systems, Information Technologies and Nanotechnologies

Kawamoto, Junpei
Department of Informatics, Faculty of Information Science and Electrical Engineering, Kyushu University

Weng, Jian
Department of Computer Science, Jinan University | Department of Informatics, Faculty of Information Science and Electrical Engineering, Kyushu University

Sakurai, Kouichi
Department of Informatics, Faculty of Information Science and Electrical Engineering, Kyushu University | Institute of Systems, Information Technologies and Nanotechnologies

出版情報：Proc. of the 6th International Conference on Trustworthy Systems, 2015. Springer International Publishing
バージョン：
権利関係：

Identity-Embedding Method for Decentralized Public-Key Infrastructure

Hiroaki Anada[1], Junpei Kawamoto[2], Jian Weng[3,2], and Kouichi Sakurai[2,1]

[1] Institute of Systems, Information Technologies and Nanotechnologies, Japan (ISIT)
{anada, sakurai}@isit.or.jp
[2] Department of Informatics, Faculty of Information Science and Electrical Engineering, Kyushu University
{kawamoto, sakurai}@inf.kyushu-u.ac.jp
[3] Department of Computer Science, Jinan University
cryptjweng@gmail.com

Abstract. A public key infrastructure (PKI) is for facilitating the authentication and distribution of public keys. Currently, the most commonly employed approach to PKI is to rely on certificate authorities (CAs), but recently there has been arising more need for decentralized peer-to-peer certification like Webs of Trust. In this paper, we propose an identity-embedding method suitable for decentralized PKI. By embedding not only ID of the candidate public-key owner itself but also IDs of his guarantors into PK, we can construct *Web of guarantors* on public keys. Here guarantors can be chosen arbitrarily by the candidate public-key owner. Our embedding method uses a combination of two public-key cryptosystems; the first cryptosystem is for PKI directly. Here we employ a technique to embed a string into a public key of the first cryptosystem. As such a string, we choose a concatenation of ID of a candidate public-key owner, IDs of his guarantors, and a public key of the second cryptosystem. This embedded public key of the second cryptosystem is used by the candidate public-key owner that he certainly knows the secret key that corresponds to the public key of the first cryptosystem. Then, with an aid of a broadcast mechanism of an updated public-key list on a peer-to-peer network, we can attain the decentralized PKI. Such an embedding method is concretely realized by the RSA encryption with the Lenstra's algorithm, which can be used as the first cryptosystem. As the second cryptosystem, we employ an elliptic curve encryption whose security is equivalent to the security of the RSA encryption, where the former achieves shorter key size than the latter. We write down concrete values of parameters for a realization of the embedding.

Keywords: identity management, public key infrastructure, decentralized system, RSA, elliptic curve.

1 Introduction

A network is for participants, so trusted identity management among participants is a must requirement. Transactions are certainly available when they

are based on reliable identities of participants; we can communicate each other, even in broadcasting, only when one party recognize the other with certainty. Concerning a trusted identity management, a public key infrastructure (PKI) is responsible for facilitating the authentication and distribution of public keys. It maintains a database of $(\mathrm{ID}, \mathrm{PK})$ pairs, where ID represents an identity, and PK represents a corresponding public key.

Currently, the most commonly employed approaches to PKIs are classified into two categories: centralized PKI with certificate authorities (CAs) and decentralized PKI of peer-to-peer certification, often referred to as Webs of Trust. CA acts a trusted third party that is responsible for distributing and managing digital certificates for a network of users. A typical example is the use of CA in the Secure Sockets Layer protocol (SSL). In the use case, an identity of a server (or a client) is an IP address and it is assured by a certificate issued. Here the certificate has, as an evidence, a digital signature generated by the certificate authority. Then the identity of the certificate authority is assured by a certificate with a signature issued by another certificate authority. Hence, there arises a chain of signatures, which forms a hierarchy of certificates with a top; that is, a root certificate. Thus the use of CAs creates single points of failure in PKI. There have been numerous recent incidents showing that too much trust is being placed in CAs. CAs have been hacked, and have even accidentally issued subordinate root certificates to customers. Additionally, while the CA system is centralized enough to introduce single points of failure, it is not centralized enough to ensure consistency. Since there exist multiple CAs, they may certify different public keys corresponding to the same identity that may yield violation of identity retention [7][4].

In contrast, in the second major PKI, Web of Trust, authentication is entirely decentralized; users are able to designate others as trustworthy by signing their public keys. A user thus accumulates a certificate containing his public key and digital signatures from entities that have assured him as trustworthy. The certificate is trusted by another user if he is able to verify that the certificate contains the signature of someone he trusts. As for motivation, Web of Trust needs relatively lower fee (in some cases, free) compared to the fee for CA-based hierarchical (centralized) certification. Another motivation of decentralized certification comes from risk control mentioned above; in a general theory of risk control, it is better to have PKI with more than one root to avoid single point failure. Ultimately saying, it is more desirable to do identity management in a *flat* manner, where flat means that *any* participant can be a guarantor. However, PGP [21] does not offer identity retention, because much like in the case of CAs there is no guarantee of consistency, and nothing prevents multiple users from creating public keys for the same identity illegally.

[4] In the research of functional encryption, there is a notion of *multi-authority* (for instance, in Lewko and Waters [15]), which means there can be more than one authorities that issue private secret keys without violation of identity retention. But in this paper, we consider this type of decentralization as introducing multiple CAs.

In this paper, we propose an identity-embedding method for decentralized PKI like Webs of Trust. Our identity-embedding method does not resolve the problem of identity retention directly, but can be used as a building block for decentralized PKI. This is because it does not need any issuing center of certificates of public keys. Instead, it needs for a participant to prove that he certainly knows the secret key to a participant, as follows.

Our decentralized PKI is on its underlying P2P network. When the network is initiated, our decentralized PKI assumes that the network has more than one participants. Each initiator generates a pair of public key and secret key and writes it into a public-key list. When a candidate public-key owner wants to take part in the underlying network, two processes are done in our decentralized PKI.

The first process is an authentication process. We use two public key cryptosystems in the first process; a candidate public-key owner generates a pair of public key and secret key of the first cryptosystem. Then, by using the second cryptosystem, the candidate public-key owner tries to prove that he certainly knows the corresponding secret key to verifiers who have been already participants. Here the verifiers are chosen as guarantors arbitrarily by the candidate public-key owner. Hence our PKI can be decentralized in a flat manner.

The second process is a broadcast process. One of guarantors adds the newly generated public key of the candidate public-key owner to the public-key list. Then the guarantor broadcasts the updated public-key list along the underlying P2P network.

The features of our decentralized PKI can be summarized in Table 1. Here, we also compare our decentralized PKI with *identity-based* PKI that has been realized with the invention of identity-based encryption [4], where any public string can be a public key.

Table 1. PKI: Centralized versus Identity-Based versus Decentralized.

Item	Centralized PKI	Identity-Based PKI	Our Decentralized PKI
Need of CA:	$\surd$	-	-
Need of KI Center:	-	$\surd$	-
PK is verified by:	Checking Certificate	PK itself	Cha.-Res. Protocol
Trust is made by:	Root CA	KI Center	Web of Guarantors

KI: Key Issuing; PK: Public Key; Cha.-Res.: Challenge-and-Response

As is stated in the above, our decentralized PKI does not need any CA. It is also notable that our decentralized PKI does not need any key issuing center, whereas identity-based PKI needs a private secret-key issuing center that yields the key-escrow problem [13, 11].

1.1 Previous Work

It is well known that there has been a history on decentralized identity management and PKI. Zimmermann, who developed PGP [21], can be considered as the pioneers of decentralized trust management. Keeping the spirit of low cost management, PGP uses a concept of a Web of Trust to establish the authenticity of the binding between a public key and its owner. Its decentralized trust model is an alternative to the centralized trust model of a public key infrastructure (PKI), which relies exclusively on a certificate authority (or a hierarchy of such).

Following Zimmermann's work, there appeared a lot of work. Blaze et al. [3] proposed a trust management system which they call PolicyMaker. Sander and Ta-Shma [18] proposed an auditable anonymous electronic cash system, whose security relies on ability of an underlying network to maintain the integrity of a public database. Concerning a digital rights management (DRM), we can see recent years several work like Qiu et al. [17], which proposed a model of social trust between content sharers of DRM-related content.

As for the Lenstra's algorithm, Lenstra [14] proposed a more efficient algorithm that allows us to embed any string I into a modulus N of the RSA encryption. The length of embeddable string I is almost the half of the bit length of N. Following this work, Kitahara et al. [13] proposed a modified algorithm to produce the two factors ($N = pq$) of the same length. Upper bounds of the length of embeddable string I have been provided by Graham and Shparlinski [9], Meng [16] and Kitahara et al. [13]. Applications have been proposed, for example, in Kitahara et al. [12].

As for a broadcast mechanism of an updated public-key list, proof-of-work is an exciting area of research [7, 2, 8]. It originates from the work of Dwork and Naor of CRYPTO '92 paper [6]. We only rely on the result of Fromknecht, Velicanu and Yakoubov [7] and Andrychowicz and Dziembowski [2], and employ their broadcast mechanisms to update a public-key list.

1.2 Our Contributions

Following the traditional spirit [21, 3, 18, 17], we contribute in two points. The first contribution is to propose an identity-embedding method for a decentralized PKI. The embedded string is taken as a concatenation of identity data of a candidate public-key owner, identity data of his guarantors, and a public key of the second cryptosystem (that is, we embed *a second public key into a first public key*). Like a digital signature, this embedding structure functions as a preventer of manipulations on public keys and hence our embedding method prevents falsification on the public-key list.

The second contribution is to provide the above identity-embedding method concretely. It is known that elliptic curve encryption with shorter public key achieves the same level of security as RSA encryption. Using this fact, we can embed a public key of an elliptic curve encryption into a modulus of an RSA

encryption. we provide a security proof, for this concrete construction of public key, that our embedding method prevents impersonation that an adversary tries to pretend an honest public-key owner without the corresponding secret key. Actually we give a security proof based on the theory of key encapsulation mechanism (KEM): our KEM is secure against adaptive chosen-plaintext attacks on one-wayness based on the Gap-CDH assumption in the random oracle model.

We note that our ID-embedding method can be used, for example, in the RSA encryption and signature in SSL. As a result, we can exit the hierarchy of certificate authorities.

Parameter values of our concrete decentralized PKI can be as follows. Here IFP denotes the Integer Factorization Problem and ECDLP denotes the Elliptic Curve Discrete Logarithm Problem. λ is the security parameter against exhaustive search, N is a modulus of the RSA encryption and p_0 is a prime order of the employed cyclic group of ECDLP. Table 2 shows parameter values.

Table 2. Parameter Size (bit).

Sec. Param. against Exhaustive Search	λ	112	128	192
Equiv. Length of Modulus for IFP	$\lambda_{\mathrm{RSA}} = \lvert N\rvert$	2048	3072	7680
Max. Length of Embed. Info.	$\lvert I\rvert = \lvert N\rvert/2 - \log_2(\lvert N\rvert) - 1$	1012	1523	3826
Equiv. Length of Order for ECDLP	$\lvert p_0\rvert = 2\ \lambda$	224	256	384
Length of Expression for a Point on EC	$\lvert P\rvert = 2\lvert p_0\rvert$	448	512	768
Room for IDs to be Embedded	$\lvert I\rvert - \lvert P\rvert$	564	1011	3058

One notable thing is that an equivalent length of prime order p_0 of a cyclic group on a elliptic curve for ECDLP is shorter than the factor p of a modulus N of the RSA encryption. Therefore we can make room to embed a public key of the elliptic curve cryptosystem into the public key of the RSA encryption.

When we use e-mail address for identity data ID, 70 characters are available because in Table 2 there are 564 bits remaining for identity data. That is, 70 byte. On condition that 1 character needs 1 byte, we can use 23 characters for identity data of a candidate public-key owner and 23 characters for identity data of two guarantors, like: `alice@decentralized.com, bob@flat.com` and `charlie@lowfee.com`.

As for the efficiency of the embedding method, the work of Kitahara et al. [13] assures that it is as efficient as the usual RSA cryptosystem in the key generation as well as encryption and decryption.

1.3 Organization of This Paper

In Section 2, we explain required notations and notions. In Section 3, we state our generic decentralized PKI. In Section 4, we describe our concrete decentralized PKI in the RSA and elliptic curve encryption setting. In Section 5, we conclude our work.

2 Preliminaries

The security parameter against exhaustive search is denoted by λ. A multiplicative cyclic group of order p_0 is denoted by $\mathbb{G}_{p_0}$. The ring of the exponent domain of $\mathbb{G}_{p_0}$, which consists of integers from 0 to $p_0 - 1$ with modulo p_0 operation, is denoted by $\mathbb{Z}_{p_0}$. When an algorithm A with input a outputs z, we denote it as $z \leftarrow A(a)$.

2.1 Embedding Technique into a Modulus of RSA Encryption

As a variant of the RSA encryption, In 1995, Vanstone and Zuccherato [19] proposed an algorithm that allows us to embed any string I into a modulus N of the RSA encryption. But it has a trade off between the time to generate the modulus N and the bit length of I. This is because that the algorithm needs factorization of I as an integer. So, when we embed I whose bit length is a half of that of N, the algorithm needs quite long time both in theory and in practice.

After that, in 1998, Lenstra [14] proposed a more efficient algorithm that allows us to embed any string I into a modulus N of the RSA encryption. The length of embeddable string I is almost the half of the bit length of N. The time to generate the modulus N is almost the same as the time to generate the modulus N of the normal RSA. The following algorithm is a modified version by Kitahara et al. [13]. Here we denote the length of a modulus N of RSA encryption that has λ-bit security against exhaustive search as λ_{RSA}.

Lenstra's Algorithm (a Modified Version [13])

1. Put $N' = I \parallel 00 \cdots 0$ s.t. $|N'| = \lambda_{\mathrm{RSA}}$.
2. Choose a prime p s.t. $|p| = \lambda_{\mathrm{RSA}}/2$ at random.
3. Compute $q' = \lceil N'/p \rceil$.
4. Compute the minimum positive integer t s.t. $q' + t$ is a prime.
5. Put $q = q' + t$.
6. Compute $N = pq$.
7. If the higher bits of N is equal to I, then return (p, q, N) else go back to 2.

Note here that (1) and (2) can be swapped.

2.2 The CDH and the Gap-CDH Problems and Assumptions

A quadruple (g, X, Y, Z) of elements in $\mathbb{G}_{p_0}$ is called a Diffie-Hellman (DH) tuple if (g, X, Y, Z) is written as (g, g^x, g^y, g^{xy}) for some elements x and y in $\mathbb{Z}_{p_0}$. A CDH problem instance is a triple $(g, X = g^x, Y = g^y)$, where the exponents x and y are random and unknown to a solver. A DDH problem instance is a quadruple (g, X, Y, Z). The DDH oracle $\mathcal{DDH}$ is an oracle which, queried about a DDH problem instance (g, X, Y, Z), replies the correct boolean decision whether (g, X, Y, Z) is a DH-tuple or not.

A CDH problem solver $\mathcal{S}$ that is allowed to access $\mathcal{DDH}$ polynomially many times is called a Gap-CDH problem solver. We define the following experiment.

$$\begin{aligned}&\mathbf{Exprmt}^{\text{gap-cdh}}_{\mathcal{S},\mathbb{G}_{p_0}}(1^\lambda)\\&x, y \leftarrow \mathbb{Z}_{p_0}, X := g^x, Y := g^y, Z \leftarrow \mathcal{S}^{\mathcal{DDH}}(g, X, Y)\\&\text{If } Z = g^{xy} \text{ then return } \textsc{Win} \text{ else return } \textsc{Lose}.\end{aligned}$$

We define the *Gap-CDH advantage of* $\mathcal{S}$ *over* **Grp** as:

$$\mathbf{Adv}^{\text{gap-cdh}}_{\mathcal{S},\mathbb{G}_{p_0}}(\lambda) \stackrel{\text{def}}{=} \Pr[\mathbf{Exprmt}^{\text{gap-cdh}}_{\mathcal{S},\mathbb{G}_{p_0}}(1^\lambda) \text{ returns } \textsc{Win}].$$

We say that the Gap-CDH Assumption holds for **Grp** if, for any PPT algorithm $\mathcal{S}$, $\mathbf{Adv}^{\text{gap-cdh}}_{\mathcal{S},\mathbb{G}_{p_0}}(\lambda)$ is negligible in λ.

2.3 Key Encapsulation Mechanism [10, 1]

A *key encapsulation mechanism (KEM)* `KEM` is a triple of PPT algorithms (**KG**, **Enc**, **Dec**). **KG** is a key generator which returns a pair of a public key and a matching secret key $(\mathrm{PK}, \mathrm{SK})$ on an input λ. **Enc** is an encapsulation algorithm which, on an input PK, returns a pair (K, ψ), where K is a random string and ψ is an encapsulation of K. **Dec** is a decapsulation algorithm which, on an input (SK, ψ), returns the decapsulation $\widehat{K}$ of ψ. We require `KEM` to satisfy the completeness condition that the decapsulation $\widehat{K}$ of a consistently generated ciphertext ψ by **Enc** is equal to the original random string K with probability one. For this requirement, we simply force **Dec** deterministic.

2.4 Adaptive Chosen Ciphertext Attack on One-Wayness of KEM

An adversary $\mathcal{A}$ performs an *adaptive chosen ciphertext attack on one-wayness* of a KEM (called one-way-CCA2, for short) in the following way [1].

$$\begin{aligned}&\mathbf{Exprmt}^{\text{ow-cca2}}_{\mathcal{A},\mathtt{KEM}}(1^\lambda)\\&(pk, sk) \leftarrow \mathbf{KG}(1^\lambda), (K^*, \psi^*) \leftarrow \mathbf{Enc}(pk)\\&\widehat{K^*} \leftarrow \mathcal{A}^{\mathcal{DEC}(sk,\cdot)}(pk, \psi^*)\\&\text{If } \widehat{K^*} = K^* \wedge \psi^* \notin \{\psi_i\}_{i=1}^{q_{dec}} \text{ then return } \textsc{Win}\\&\text{else return } \textsc{Lose}.\end{aligned}$$

In the above experiment, $\psi_i, i = 1, \ldots, q_{dec}$ mean ciphertexts for which $\mathcal{A}$ queries its decapsulation oracle $\mathcal{DEC}(sk, \cdot)$ for the answers. Here the number q_{dec} of queries is polynomial in k. Note that the challenge ciphertext ψ^* itself must not be queried to $\mathcal{DEC}(sk, \cdot)$, as is described $\psi^* \notin \{\psi_i\}_{i=1}^{q_{dec}}$ in the experiment.

We define *the one-way-CCA2 advantage of* $\mathcal{A}$ *over* `KEM` as:

$$\mathbf{Adv}^{\text{ow-cca2}}_{\mathcal{A},\mathtt{KEM}}(\lambda) \stackrel{\text{def}}{=} \Pr[\mathbf{Exprmt}^{\text{ow-cca2}}_{\mathcal{A},\mathtt{KEM}}(1^\lambda) \text{ returns } \textsc{Win}].$$

We say that a KEM is secure against adaptive chosen ciphertext attacks against one-wayness (one-way-CCA2-secure, for short) if, for any PPT algorithm $\mathcal{A}$, $\mathbf{Adv}^{\text{ow-cca2}}_{\mathcal{A},\mathtt{KEM}}(\lambda)$ is negligible in k.

Note that if a KEM is IND-CCA2 secure [5], then it is one-way-CCA2 secure. So IND-CCA2 security is a stronger notion than one-way-CCA2 security.

3 Our Generic Description of Embedding Method and Decentralized PKI

In this section, we describe our generic embedding method and a related decentralized PKI. We first state an assumption for the underlying network. Next, we explain a design principle of our embedding method. Then, we describe a related decentralized PKI.

Assumption for Underlying Network Our decentralized PKI utilizes a public-key list that is public to the underlying network. The public-key list should be examined and maintained by all participants who are active in the network. The security of our decentralized PKI will partially rely on the ability of an underlying network to maintain the integrity of the public-key list.

3.1 Components and Procedures of Our Generic Decentralized PKI

Initiation We start with at least n initiators. n must be equal to the number of guarantors for a candidate public-key owner. Each initiator generates a pair of public key and secret key and writes it to a public-key list.

Generation of Candidate Public-Key Owner's New Key When a candidate public-key owner wants to join the underlying network, he executes the following.

1. Generate a secret key sk_0 by running $\mathbf{KG}(\lambda)$.
2. Compute a value of the one-way function at sk_0: $\overline{sk_0} := f(sk_0)$.
3. Put $I = \mathrm{ID}_{\mathrm{cand}} \parallel \mathrm{ID}_{\mathrm{grnt}_1} \parallel \cdots \parallel \mathrm{ID}_{\mathrm{grnt}_n} \parallel \overline{sk_0}$.
4. Put $pk' = I \parallel 00\cdots0$.
5. Apply the embedding algorithm to pk' to obtain (pk, sk).
6. Put $\mathrm{PK}_{\mathrm{cand}} = pk$, $\mathrm{SK}_{\mathrm{cand}} = sk$.

Identification of Candidate Public-Key Owner

1. A verifier generates a random challenge according to a challenge-and-response identification protocol and send it to the candidate public-key owner.
2. Receiving the random challenge, the candidate public-key owner generates a response according to the protocol, and send it to the verifier.
3. Receiving the response, the verifier verifies it and outputs `accept` or `reject`.

The above protocol is executed by all guarantors, $i = 1, \ldots, n$, and any participant who wants to verify the identity of other participant with whom the former communicates.

Local Update of Public-Key List After finishing the above verification, one of the guarantors adds the newly generated public key $\mathrm{PK}_{\mathrm{cand}}$ of the new participant to the public-key list.

Broadcast of the Updated Identity List The same guarantor broadcasts the updated public-key list along the underlying network. Integrity of the public-key list is assured by the above assumption.

4 Instantiation

In this section, we instantiate our generic decentralized PKI by employing the RSA encryption as the first cryptosystem and the elliptic curve encryption as the second cryptosystem. We use the modified version [13] of the Lenstra's algorithm [14] as our main tool.

4.1 Components and Procedures of Our Decentralized PKI: Instantiation

We assume that the Elliptic Curve Discrete-Logarithm problem for the employed cyclic group $\mathbb{G}_{p_0}$ of prime order p_0 and the Integer-Factorization problem for the employed RSA modulus N have almost the same difficulty ([20]).

Initiation This phase is executed generically according to the description in Section 3.1.

Generation of Candidate Public-Key Owner's New Key

1. Generate a prime p s.t. $|p| = \lambda_{\mathrm{RSA}}/2$ at random.
2. Compute $P := g^p$ in $\mathbb{G}_{p_0}$.
3. Put $I = \mathrm{ID}_{\mathrm{cand}} \parallel \mathrm{ID}_{\mathrm{grnt}_1} \parallel \cdots \parallel \mathrm{ID}_{\mathrm{grnt}_n} \parallel P$.
4. Put $N' = I \parallel 00 \cdots 0$ s.t. $|N'| = \lambda_{\mathrm{RSA}}$.
5. Apply the modified Lensra's algorithm to obtain (N, q) and (e, d) s.t. $ed = 1 \bmod \phi(N)$.
6. Put $\mathrm{PK}_{\mathrm{cand}} = (N, e)$, $\mathrm{SK}_{\mathrm{cand}} = (q, d)$.

Identification of Candidate Public-Key Owner

1. The verifier chooses a random exponent t from $\mathbb{Z}_{p_0}$, computes $h = P^t$ in $\mathbb{G}_{p_0}$, and send it to the candidate public-key owner.
2. Receiving the random challenge h, the candidate public-key owner computes $K' = h^q$ by using his secret key q, then compute its hash value $K = H_\mu(K)$ a response, and then send it to the verifier.
3. Receiving the response K, the verifier verifies it by examining the following equation, and outputs `accept` or `reject`accordingly.

$$K \stackrel{?}{=} H_\mu(g^{Nt}).$$

The above protocol is executed by all guarantors, $i = 1, \ldots, n$, and any participant who wants to verify the identity of other participant with whom the former communicates.

Correctness of the above protocol is assured by:

$$K = H_\mu(K') = H_\mu((P^t)^q) = H_\mu((P^q)^t) = H_\mu(((g^p)^q)^t) = H_\mu(g^{Nt}).$$

Note that we can view the above procedures as a hashed key encapsulation mechanism [10, 1], `h-EGKEM`, via putting $g := P$, $x := q$ and $X = g^x = P^q := g^N$ in its algorithm described in Fig. 1.

Note that, for a prime q (a factor of the RSA modulus N) that is longer than p_0 (the equivalent prime order of the group $\mathbb{G}_{p_0}$ of the elliptic curve cryptography), *collision resistance of the secret keys* $x = q \bmod p_0$ is assured by *the Dirichlet's Theorem on Primes in Arithmetic Progressions*.

Note also that, if a candidate candidate public-key owner generate a modulus N illegally, say, $N = pqr$ (three factors), the candidate public-key owner merely weaken his security of RSA.

Key Generation
• **KG**: given λ as an input;
– $(p_0, \mathbb{G}_{p_0}, g) \leftarrow \mathbf{Grp}(\lambda), x \leftarrow \mathbb{Z}_{p_0}, X := g^x$
– $\mathrm{PK}_0 := (g, X), \mathrm{SK}_0 := (g, x)$, return $(\mathrm{PK}_0, \mathrm{SK}_0)$

Encapsulation
• **Enc**: given PK_0 as an input;
– $a \leftarrow \mathbb{Z}_{p_0}, K' := X^a, K := H_\mu(K'), h := g^a, \psi := h$, return (K, ψ)

Decapsulation
• **Dec**: given SK_0 and $\psi = h$ as an input;
– $\widehat{K'} := h^x$
– $\widehat{K} := H_\mu(\widehat{K'})$, return $\widehat{K}$

Fig. 1. Our Hashed ElGamal KEM: `h-EGKEM`.

Local Update and Broadcast of Public-Key List These phases are executed generically according to the description in Section 3.1.

4.2 Attack and Security in Our Instantiation

An attack to be considered on the above verification procedure is an impersonation by a cheating verifier. More precisely, in the phase that an honest public-key owner tries to prove that he knows a factoring of N to an honest verifier, a man-in-the-middle adversary can execute impersonation. In our case, the security against this attack is reduced to the one-way-CCA2 security of our `h-EGKEM`.

Theorem 1 *If the key encapsulation mechanism* ***h-EGKEM*** *is one-way-CCA2 secure, then the protocol of identification of a candidate public-key owner is secure against man-in-the-middle attacks of impersonation.*

Proof. Suppose that there is a PPT, man-in-the-middle adversary $\mathcal{M}$. Then, putting $g := P$ and $X := g^N$, we can make a PPT adversary that attacks on `h-EGKEM` and that has the same success probability. □

Theorem 2 *The key encapsulation mechanism* ***h-EGKEM*** *is one-way-CCA2 secure based on the Gap-CDH assumption for* **Grp** *in the random oracle model. More precisely, for any PPT one-way-CCA2 adversary $\mathcal{A}$ on* ***h-EGKEM****, and assuming that $\mathcal{A}$ issues a hash query every time when $\mathcal{A}$ computes a hash value, there exist a PPT Gap-CDH problem solver $\mathcal{S}$ on* **Grp***which satisfies the following tight reduction.*

$$\mathbf{Adv}^{ow\text{-}cca2}_{\mathcal{A},\textit{h-EGKEM}}(\lambda) \leqslant \mathbf{Adv}^{gap\text{-}cdh}_{\mathcal{S},\mathbb{G}_{p_0}}(\lambda).$$

Proof. Employing any given adversary $\mathcal{A}$ on `h-EGKEM` as subroutine, we construct, in the random oracle model, a PPT Gap-CDH problem solver $\mathcal{S}$ as follows (see Fig.2).

$\mathcal{S}$ is given a CDH problem instance (g, X, Y) as input. $\mathcal{S}$ initialize the hash table T, whose row consists of the format (h, K', K). $\mathcal{S}$ sets $\mathrm{PK}_0 := X$ and $\phi : * := Y$, where the latter is the challenge message that should be responded by $\mathcal{A}$. $\mathcal{S}$ invokes $\mathcal{A}$ on input $\mathrm{PK}_0 := X$ and $\phi : *$.

$\mathcal{S}$ must answer decapsulation queries and hash queries of $\mathcal{A}$. Those answers can be made as is in the Fig.2.

Finally, when $\mathcal{A}$ responds an answer $\widehat{K}^*$, $\mathcal{S}$ can extract the answer $Z := K'$ of the CDH problem instance (g, X, Y). □

4.3 Discussion

Multiple Identities As we can see from our procedures, There is a possibility that more than one public key N are issued on the same $\mathrm{ID}_{\mathrm{cand}}$. That is, N_1 and N_2 $(N_1 \neq N_2)$ both have the same $\mathrm{ID}_{\mathrm{cand}}$. It would be desirable if guarantors could control this phenomenon.

Given (g, X, Y) as input;

Initial Setting
– Initialize the hash table T
– $\mathrm{PK}_0 := (g, X)$, $\psi^* := Y$
– Invoke $\mathcal{A}$ on (PK_0, ψ^*)

Answering $\mathcal{A}$'s Queries and Extracting the Answer

• When $\mathcal{A}$ queries its decap. oracle $\mathcal{DEC}(\mathrm{SK}, \cdot)$ for the answer of $\psi = h$;
– If $\psi = \psi^*$, then $\widehat{K} := \perp$
– else if h is in T, then pick K in the same row, then $\widehat{K} := K$
– else if h is not in T, then search K' s.t. $\mathcal{DDH}(g, X, h, K') = 1$
– – If there is such K', then pick K in the same row, $\widehat{K} := K$, $T := T \cup \{(h, K', K)\}$
– – else $K' \stackrel{\$}{\leftarrow} \mathbb{G}_{p_0}$, $\widehat{K} := K := H(K')$, $T := T \cup \{(h, K', K)\}$
– Reply $\widehat{K}$ to $\mathcal{A}$

• When $\mathcal{A}$ queries its hash oracle $H(\cdot)$ for the hash value of K';
– If K' is in T, then pick K in the same row
– else if K' is not in T, search h s.t. $\mathcal{DDH}(g, X, h, K') = 1$
– – If there is such h, then pick K in the same row, $T := T \cup \{(h, K', K)\}$
– – else $K' \stackrel{\$}{\leftarrow} \mathbb{G}_{p_0}$, $K := H(K')$, $T := T \cup \{(h, K', K)\}$
– Reply K to $\mathcal{A}$

• When $\mathcal{A}$ responds $\widehat{K}^*$;
– Search K in T s.t. $K = \widehat{K}^*$, pick K' in the same row
– Return $Z := K'$

Fig. 2. A Gap-CDH Problem Solver $\mathcal{S}$ for the Proof of Theorem 2.

Revocation As is discussed for PGP and a web of trust, our decentralized PKI also has the problem revocation. A simple way to enable revocation is to maintain a revocation list as well as our public-key list. We have to rely on an assumption that the underlying network can maintain the integrity of the revocation list, too.

Anonymity of Guarantors In the real world it might be better to anonymise guarantors. This is a matter to be pursued. Using an anonymous credential system can be considered to resolve this matter though it needs a (centralized) issuer of credentials.

5 Conclusions

We proposed an embedding method for a decentralized PKI; ID of a candidate public-key owner, IDs of his guarantors, and a public key of the second cryptosystem was embedded into a public key of the first cryptosystem. This

embedding functions as a preventer of manipulations on public keys. But it prevents not only falsification but also impersonation. We realized our decentralized PKI concretely; we could embed a public key of an elliptic curve encryption into a modulus of an RSA encryption. We note that the resulting decentralized PKI can be used as an alternative of the RSA encryption and signature in SSL.

6 Acknowledgements

The third author was partially supported by the Invitation Programs for Foreign-based Researchers provided by the National Institute of Information and Communications Technology (NICT), Japan.

The first, second and forth authors were partially supported by the Bilateral Joint Research Projects/Seminars FY2014 by Japan Society for the Promotion of Science under the research project name "Computational Aspects of Mathematical Design and Analysis of Secure Communication Systems Based on Cryptographic Primitives", who appreciate sincere thanks for discussion with Sushmita Ruj in Indian Statistical Institute and Avishek Adhikari in University of Calcutta.

References

1. H. Anada and S. Arita. Identification Schemes from Key Encapsulation Mechanisms. In *AFRICACRYPT*, volume 6737, pages 59–76, 2011.
2. M. Andrychowicz and S. Dziembowski. Distributed Cryptography Based on the Proofs of Work. Cryptology ePrint Archive, Report 2014/796, 2014. http://eprint.iacr.org/.
3. M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized Trust Management. In *In Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 164–173. IEEE Computer Society Press, 1996.
4. D. Boneh and M. K. Franklin. Identity-Based Encryption from the Weil Pairing. In *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, pages 213–229, 2001.
5. R. Cramer and V. Shoup. A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack. In *CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer, 1998.
6. C. Dwork and M. Naor. Pricing via Processing or Combatting Junk Mail. In *Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '92, pages 139–147, London, UK, UK, 1993. Springer-Verlag.
7. C. Fromknecht, D. Velicanu, and S. Yakoubov. A Decentralized Public Key Infrastructure with Identity Retention. Cryptology ePrint Archive, Report 2014/803, 2014. http://eprint.iacr.org/.
8. C. Garman, M. Green, and I. Miers. Decentralized Anonymous Credentials. *IACR Cryptology ePrint Archive*, 2013:622, 2013.
9. S. W. Graham and I. E. Shparlinski. On RSA moduli with almost half of the bits prescribed. *Discrete Applied Mathematics*, 156(16):3150–3154, 2008.

10. E. Kiltz. Chosen-Ciphertext Security from Tag-Based Encryption. In *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, pages 581–600, 2006.
11. M. Kitahara, H. Anada, J. Kawamoto, and K. Sakurai. Embedding Method of Owner's Information into Public Key of RSA Encryption and its Application to Digital Rights Management System. In *IPSJ SIG Technical Report*, volume Vol.2014-CSEC65, page No.3. Information Processing Society of Japan, 2014.
12. M. Kitahara, T. Nishide, and K. Sakurai. A Method for Embedding Secret Key Information in RSA Public Key and Its Application. In *Proceedings of the Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 665–670. IEEE, 2012.
13. M. Kitahara, T. Yasuda, T. Nishide, and K. Sakurai. Upper Bound of the Length of Information Embedd in RSA Public Key Efficiently. In *AsiaPKC@AsiaCCS*, pages 33–38. ACM, 2013.
14. A. K. Lenstra. Generating RSA Moduli with a Predetermined Portion. In *Proceedings of the International Conference on the Theory and Applications of Cryptology and Information Security: Advances in Cryptology*, ASIACRYPT '98, pages 1–10, London, UK, UK, 1998. Springer-Verlag.
15. A. Lewko and B. Waters. Decentralizing Attribute-based Encryption. In *Proceedings of the 30th Annual International Conference on Theory and Applications of Cryptographic Techniques: Advances in Cryptology*, EUROCRYPT'11, pages 568–588, Berlin, Heidelberg, 2011. Springer-Verlag.
16. X. Meng. On RSA moduli with half of the bits prescribed. *Journal of Number Theory*, 133(1):105–109, 2013.
17. Q. Qiu, Z. Tang, F. Li, and Y. Yu. A Personal DRM Scheme Based on Social Trust. *Chinese Journal of Electronics*, 21(4):719–724, 2012.
18. T. Sander and A. Ta-Shma. Auditable, Anonymous Electronic Cash. In *CRYPTO*, LNCS, pages 555–572. Springer, 1999.
19. S. A. Vanstone and R. J. Zuccherato. Short RSA Keys and Their Generation. *J. Cryptology*, 8(2):101–114, 1995.
20. M. Yasuda, T. Shimoyama, J. Kogure, and T. Izu. On the Strength Comparison of the ECDLP and the IFP. In I. Visconti and R. D. Prisco, editors, *SCN*, volume 7485 of *Lecture Notes in Computer Science*, pages 302–325. Springer, 2012.
21. Zimmermann. Phil zimmermann & associates llc. http://www.philzimmermann.com/EN/background/index.html, 2014. [Online; accessed 20-Sept-2014].