分散型アイデンティティ管理スキームと そのRSA及び離散対数系暗号による実現

穴田 啓晃† 川本 淳平‡ 翁 健∗‡ 櫻井 幸一‡†

† 公益財団法人九州先端科学技術研究所
814-0002 福岡市早良区 2-1-22 福岡 SRP センタービル 7 階
anada@isit.or.jp

‡ 九州大学大学院システム情報科学研究院
819-0395 福岡市西区元岡 744 番地 ウェスト 2 号館 712
{kawamoto,sakurai}@inf.kyushu-u.ac.jp

∗ 暨南大学計算機科学科
510632 中国広東省広州市黄埔大道西 601 号
cryptjweng@gmail.com

あらまし ネットワークの参加者のアイデンティティ管理は通常，中央権限者によってなされる．本論文で我々は，公開鍵暗号に基づく分散型のアイデンティティ管理スキームを提案する．我々は参加候補者の公開鍵の中に任意の（限られた長さの）ストリングを埋め込む技術を用いる．埋め込むストリングとしては参加候補者やその保証人のアイデンティティデータ，及び，参加候補者が確かにその公開鍵及び対応する秘密鍵を生成したことを証明するためのもう一つの公開鍵を取る．我々はこの埋め込み手法を，アイデンティティリストを更新しブロードキャストする機構と結びつけ，分散型のアイデンティティ管理スキームを構成する．この埋め込み手法は，RSA 暗号における Lenstra のアルゴリズム及びその改良変種によって実現される．我々は具体的なパラメータを書き下し，大きさを評価する．

Decentralized Identity Management Scheme and its Realization by RSA and Discrete-Log-Based Encryption

Hiroaki Anada† Junpei Kawamoto‡ Jian Weng∗‡ Kouichi Sakurai‡†

†Institute of Systems, Information Technologies and Nanotechnologies (ISIT)
Fukuoka SRP Center Building 7F, 2-1-22, Momochihama, Sawara-ku, Fukuoka, 814-0001, JAPAN
anada@isit.or.jp

‡Faculty of Information Science and Electrical Engineering, Kyushu University
W2-712, Motooka 744, Nishi-ku, Fukuoka-city, 819-0395, JAPAN
{kawamoto,sakurai}@inf.kyushu-u.ac.jp

∗ Department of Computer Science, Jinan University
601 Huangpu W Ave, Tianhe, Guangzhou, Guangdong, 510632, CHINA
cryptjweng@gmail.com

Abstract Identity management of a network participants is usually done by a centralized authority. In this paper, we propose a decentralized identity management scheme based on a public-key cryptosystem. We assume that there is a technique to embed any string (of limited length) into a public key. Then, as the string, we choose identity data of a candidate participant and his guarantors as well as another public key which is used to prove that the candidate participant certainly generate the public key and the corresponding secret key. Combining the embedding method with a broadcast mechanism of an updated identity list, we can attain an identity management scheme in a decentralized manner. Such embedding method is realized by the Lenstra's algorithm in the RSA encryption and its modified variant. We write down concrete parameters for the realization and evaluate the size of parameters and computational efficiency.

1 Introduction

A network is for participants, so trust among them is a must requirement. Transactions are certainly available when they are based on the trust. A kind of trust is reliability of identities of participants; we can communicate each other in the network, even in broadcasting, when we can recognize the participant with whom we are communicating. Hence identity management has an importance and to prevent impersonation is necessary. Such reliable identity management until now is mainly done by a centralized authority, but there have arised needs to do the management in a decentralized, flat manner. One motivation for the decentralized management is the Peer-to-Peer (P2P) communication. P2P communication can make transactions available with relatively lower fee comapared to the Server-Client (centralized) communication.

1.1 Previous Work

There was a history concerning decentralized identity management. Zimmermann, who developedd PGP [13], is one of the pioneers of decentralized trust management. The spirit of PGP is P2P low cost management. Blaze et al. [2] proposed a trust management system which they call PolicyMaker. Concerning a digital rights management (DRM), we can see recent years an ingredient that Sander and Ta-Shma [10] proposed an auditable anonymous electronic cash system, whose security relies on ability of an underlying network to maintain the integrity of a public database. QIU et al. [9] proposed a model of social trust between content sharers of DRM-related content.

1.2 Our Contributions

According to traditional challenges [13, 2, 10, 9], our first contribution is to propose a decentralized identity management scheme by embedding identity strings into a public key of an encryption scheme.

Our second contribution is to provide the above decentralized identity management scheme in a concrete way. By embedding another public key of an empolyed discrete-logarithm based encryption into a modulus of the RSA encryption. Our scheme is secure even if the security of the empolyed discrete-logarithm based encryption collapses.

1.3 Organization of This Paper

In Section 2, we explain required notations and notions. In Section 3, we state our generic identity management scheme. In Section 4, we describe our concrete scheme in the RSA and discrete logarithm setting. In Section 5, we conclude our work.

2 Preliminaries

The security parameter is denoted by λ. A prime of bit length λ is denoted by p. A multiplicative cyclic group of order p is denoted by $\mathbb{G}$. The ring of the exponent domain of $\mathbb{G}$, which consists of integers from 0 to $p-1$ with modulo p operation, is denoted by $\mathbb{Z}_p$. 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 [11] 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 emebed 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 [7] proposed a more efficient algorithm that allows us to embed any string I whose bit length is a half of that of N into a modulus N of the RSA encryption. 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. [6].

Lenstra's Algorithm (a Modified Version [6])

1. Put $N' = I \parallel 00\cdots0$ s.t. $|N'| = \lambda$.

2. Choose a prime p s.t. $|p| = \lambda/2$ 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 Gap-DL Problem and Assumption

A discrete log (DL) problem instance consists of $(g, X = g^x)$, where the exponent x is random and unknown to a solver. A DL problem solver is a PPT algorithm which, given a random DL problem instance (g, X) as an input, tries to return x.

A quadruple (g, X, Y, Z) of elements in $\mathbb{G}$ 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. The CDH oracle $\mathcal{CDH}$ is an oracle which, queried about a CDH problem instance (g, X, Y), replies the correct answer $Z = g^{xy}$.

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

$\mathbf{Exprmt}^{\text{gap-dl}}_{\mathcal{S},\mathbf{Grp}}(\lambda)$
$(p, g) \leftarrow \mathbf{Grp}(\lambda), x \leftarrow \mathbb{Z}_{p_0}, X := g^x$
$x^* \leftarrow \mathcal{S}^{\mathcal{CDH}}(g, X)$
If $g^{x^*} = X$ then return Win else return Lose.

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

$$\mathbf{Adv}^{\text{gap-dl}}_{\mathcal{S},\mathbf{Grp}}(\lambda) \stackrel{\text{def}}{=} \Pr[\mathbf{Exprmt}^{\text{gap-dl}}_{\mathcal{S},\mathbf{Grp}}(\lambda) \text{ returns Win}].$$

We say that the Gap-DL Assumption holds for **Grp** if, for any PPT algorithm $\mathcal{S}$, $\mathbf{Adv}^{\text{gap-dl}}_{\mathcal{S},\mathbf{Grp}}(\lambda)$ is negligible in k.

Although the Gap-DL Assumption is considered fairly strong, it is believed to hold at least for a certain class of cyclic groups [8].

2.3 The Knowledge-of-Exponent Assumption

Informally, the Knowledge-of-Exponent Assumption (KEA) [5, 1] says that, given a randomly chosen $h \in \mathbb{G}$ as an input, a PPT algorithm $\mathcal{H}$ can extend (g, h) to a DH-tuple (g, h, X, Z) *only when* $\mathcal{H}$ *knows the exponent* x *of* $X = g^x$. The formal definition is described as follows.

Let $\Lambda(\lambda)$ be any distribution. Let $\mathcal{H}$ and $\mathcal{H}'$ be any PPT algorithms which take input of the form (g, h, λ). Here g is any fixed generator, h is a randomly chosen element in $\mathbb{G}$, and λ is a string in $\{1, 0\}^*$ output by $\Lambda(\lambda)$ called auxiliary input [3, 4]. We define the following experiment.

$\mathbf{Exprmt}^{\text{kea}}_{\mathcal{H},\mathcal{H}',\mathbf{Grp}}(\lambda)$
$(p, g) \leftarrow \mathbf{Grp}(\lambda), \lambda \leftarrow \Lambda(\lambda), a \leftarrow \mathbb{Z}_{p_0}, h := g^a$
$(g, h, X, Z) \leftarrow \mathcal{H}(g, h, \lambda), x' \leftarrow \mathcal{H}'(g, h, \lambda)$
If $X^a = Z \wedge g^{x'} \neq X$ then return Win
else return Lose.

Note that λ is independent of h in the experiment. This independence is crucial ([3, 4]).

We define the *KEA advantage of* $\mathcal{H}$ *over* **Grp** *and* $\mathcal{H}'$ as:

$$\mathbf{Adv}^{\text{kea}}_{\mathcal{H},\mathcal{H}',\mathbf{Grp}}(\lambda) \stackrel{\text{def}}{=} \Pr[\mathbf{Exprmt}^{\text{kea}}_{\mathcal{H},\mathcal{H}',\mathbf{Grp}}(\lambda) \text{ returns Win}].$$

An algorithm $\mathcal{H}'$ is called the *KEA extractor*. $\mathbf{Adv}^{\text{kea}}_{\mathcal{H},\mathcal{H}',\mathbf{Grp}}(\lambda)$ can be considered the probability that the KEA extractor $\mathcal{H}'$ *fails* to extract the exponent x of $X = g^x$. We say that KEA holds for **Grp** if, for any PPT algorithm $\mathcal{H}$, there exists a PPT algorithm $\mathcal{H}'$ such that for any distribution $\Lambda(\lambda)$ $\mathbf{Adv}^{\text{kea}}_{\mathcal{H},\mathcal{H}',\mathbf{Grp}}(\lambda)$ is negligible in k.

2.4 Key Encapsulation Mechanism

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.

2.4.1 Non-adaptive Chosen Ciphertext Attack on One-Wayness of KEM

Suppose that an adversary $\mathcal{A}$ consists of two algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$. The following experiment is called a *non-adaptive chosen ciphertext attack on one-wayness* of a KEM (called one-way-CCA1, for short).

$\mathbf{Exprmt}^{\text{ow-cca1}}_{\mathcal{A},\mathtt{KEM}}(\lambda)$
$(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathbf{KG}(\lambda), st \leftarrow \mathcal{A}_1^{\mathcal{DEC}(\mathrm{SK},\cdot)}(\mathrm{PK})$
$(K^*, \psi^*) \leftarrow \mathbf{Enc}(\mathrm{PK}), \widehat{K^*} \leftarrow \mathcal{A}_2(st, \psi^*)$
If $\widehat{K^*} = K^*$ then return WIN else return LOSE.

We define *the one-way-CCA1 advantage of $\mathcal{A}$ over KEM* as:

$$\mathbf{Adv}^{\text{ow-cca1}}_{\mathcal{A},\mathtt{KEM}}(\lambda) \stackrel{\text{def}}{=} \Pr[\mathbf{Exprmt}^{\text{ow-cca1}}_{\mathcal{A},\mathtt{KEM}}(\lambda) \text{ returns WIN}].$$

We say that an KEM is secure against non-adaptive chosen ciphertext attacks (one-way-CCA1-secure, for short) if, for any PPT algorithm $\mathcal{A}$, $\mathbf{Adv}^{\text{ow-cca1}}_{\mathcal{A},\mathtt{KEM}}(\lambda)$ is negligible in k.

The non-adaptive chosen ciphertext attack is a stronger model than the passive attack because $\mathcal{A}_1$ can choose ciphertexts of his choice in its queries to $\mathcal{DEC}$.

3 Our Generic Identity Management Scheme

In this section, we describe our generic identity management scheme. we first state an assumption for the underlying network. Next, we explain a design principle of our identity management scheme. Then, we describe components and procedures of our scheme.

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

3.1 Design Principle, Components and Procedures of Our Generic Scheme

3.1.1 Initiation

We start with at least n initiators. n must be equal to the number of guarantors for a candidate participant.

3.1.2 Local Update of the ID-list

When a new, candidate participant comes, the candidate participant issues a query that he wants to join the network.

3.1.3 Generation of Candidate-Participant's New Key

1. Generate a secret key $\mathrm{SK} = (sk_1)$ by running $\mathbf{KG}(\lambda, 1)$.
2. Compute a value of the one-way function $\overline{sk_1} := f(sk_1)$.
3. Put $I = \mathrm{ID}_{\mathrm{cand}} \parallel \mathrm{ID}_{\mathrm{gurt}_1} \parallel \cdots \parallel \mathrm{ID}_{\mathrm{gurt}_n} \parallel \overline{sk_1}$.
4. Put $pk' = I \parallel 00 \cdots 0$.
5. Apply the embedding algorithm to pk' to obtain (pk, sk).
6. Put $\mathrm{PK}_{\mathrm{cand}} = pk$, $\mathrm{SK}_{\mathrm{cand}} = sk$.

3.1.4 Verification of Candidate-Participant's New Key

1. The i-th guarantor generates a random challenge according to a challenge-and-response identification protocol and send it to the candidate-participant.

2. Receiving the random challenge, the candidate-participant generates a response according to the protocol, and send it to the i-th guarantor.
3. Receiving the response, the i-th guarantor verifies it and outputs `accept` or `reject`.
4. The above protocol is executed by all guarantors; $i = 1, \ldots, n$.

3.1.5 Broadcast of the Updated ID-list

1. After finishing the above verification, the candidate-participant adds his identity data $\mathrm{ID}_{\mathrm{cand}}$ on the ID-list A broadcast mechanism for an updated identity list starts. Integrity of the identity list is assured by the above assumption.

4 Instantiation

In this section, we instantiate our generic identity management scheme by public-key encryptions in RSA and discrete logarithm setting. We use the modified version [6] of the Lenstra's algorithm [7] as our main tool.

4.1 Components and Procedures of Our Scheme in RSA Setting

We assume that the Elliptic Curve Discrete-Logarithm problem for the group $\mathbb{G}$ and the Integer-Factorization problem for the RSA modulus N have almost the same difficulty ([12]).

4.1.1 Generation of Candidate-Participant's New Key

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

4.1.2 Verification of Candidate-Participant's New Key

1. The i-th guarantor chooses a random exponent t from $\mathbb{Z}_{p_0}$, computes $h = P^t$ in $\mathbb{G}$, and send it to the candidate-participant.
2. Receiving the random challenge h, the candidate-participant computes a response value $K = h^q$ by using his secret key q, and send it to the i-th guarantor.
3. Receiving the response, the i-th guarantor verifies by using the following equation, and outputs `accept` or `reject`.

$$K \stackrel{?}{=} g^{Nt}.$$

4. The above protocol is executed by all guarantors; $i = 1, \ldots, n$.

Correctness of the above protocol is assured by:

$$K = (P^t)^q = (P^q)^t = ((g^p)^q)^t = g^{Nt}.$$

Note that we can view the above procedures as a key encapsulation mechanism of El Gamal encryption by putting $g := P$ and $x := q$ in the algorithm in Fig. 1.

Note also that *the hardness of finding the secret key $x = q$* for the employed cyclic group $\mathbb{G}$ is assured by the hardness of Integer Factorization Problem for the modulus N of the RSA encryption.

4.1.3 Broadcast of the Updated ID-list

This phase is executed generically according to the description in Section 3.1.5.

4.2 Attack and Security on Our Scheme in RSA Setting

Theorem 1 *The key encapsulation mechanism* `EGKEM` *is one-way-CCA1 secure based on the Gap-DL assumption and KEA for* **Grp**. *More precisely, for*

Key Generation

– **KG**: given λ as an input;

• $(p_0, g) \leftarrow \mathbf{Grp}(\lambda), x \leftarrow \mathbb{Z}_{p_0}, X := g^x$

• $\mathrm{PK}_0 := (p_0, g, X), \mathrm{SK}_0 := (p_0, g, x)$, return $(\mathrm{PK}_0, \mathrm{SK}_0)$

Encapsulation

– **Enc**: given PK_0 as an input;

• $a \leftarrow \mathbb{Z}_{p_0}, K := X^a, h := g^a, \psi := h$, return (K, ψ)

Decapsulation

– **Dec**: given SK_0 and $\psi = h$ as an input;

• $\widehat{K} := h^x$, return $\widehat{K}$

図 1: El Gamal KEM: EGKEM.

any PPT one-way-CCA1 adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *on* EGKEM*, there exist a PPT Gap-DL problem solver* $\mathcal{S}$ *on* **Grp** *and a PPT algorithm* $\mathcal{H}$ *for KEA which satisfy the following tight reduction.*

$$\mathbf{Adv}^{ow\text{-}cca1}_{\mathcal{A},EGKEM}(\lambda) \leqslant \mathbf{Adv}^{gap\text{-}dl}_{\mathcal{S},\mathbf{Grp}}(\lambda) + \mathbf{Adv}^{kea}_{\mathcal{H},\mathcal{H}',\mathbf{Grp}}(\lambda).$$

Proof. An outline is stated as follows. The CDH oracle enables the solver $\mathcal{S}$ to simulate the adversary $\mathcal{A}$'s decapsulation oracle perfectly. After that the KEA extractor works to extract the answer of a DL problem instance. (Note that the Gap-DL assumption and KEA are compatible.)

Let us proceed in detail. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be as in Theorem. Using $\mathcal{A}$ as a subroutine, we construct a Gap-DL problem solver $\mathcal{S}$ as follows. (The construction is illustrated in Fig.2.)

$\mathcal{S}$ is given $p_0, g, X = g^x$ as a DL problem instance, where x is random and unknown to $\mathcal{S}$. $\mathcal{S}$ initializes its inner state, sets $pk = (p_0, g, X)$ and invokes $\mathcal{A}_1$ on an input pk.

In the first phase, in the case that $\mathcal{A}_1$ queries its decapsulation oracle $\mathcal{DEC}(sk, \cdot)$ for the answer for $\psi = h$, $\mathcal{S}$ queries its CDH oracle $\mathcal{CDH}$ for the answer for a CDH problem instance (g, h_0, X) and gets Z as a reply. Then, $\mathcal{S}$ replies $\widehat{K} = Z$ to $\mathcal{A}$. In the case that $\mathcal{A}_1$ returns the inner state st, $\mathcal{S}$ ends the first phase and proceeds to the second phase.

In the second phase, $\mathcal{S}$ chooses a^* from $\mathbb{Z}_{p_0}$ at random and computes $\psi^* = h^* = g^{a^*}$. Then, $\mathcal{S}$ invokes $\mathcal{A}_2$ on an input (st, ψ^*). In the case that $\mathcal{A}_2$ returns $\widehat{K}^*$, $\mathcal{S}$ invokes the KEA extractor $\mathcal{H}'$ on (g, h^*, st). Here $\mathcal{H}'$ is the KEA extractor associated with the algorithm $\mathcal{H}$ below.

$\mathcal{H}(g, h^*, st)$:

$\widehat{K}^* \leftarrow \mathcal{A}_2(st, h^*), Z := \widehat{K}^*, \mathrm{return}(g, h^*, X, Z).$

Note that the auxiliary input st is independent of h^*.

If $\mathcal{H}'$ returns x^*, then $\mathcal{S}$ returns x^*.

It is obvious that $\mathcal{S}$ simulates A's decapsulation oracle $\mathcal{DEC}(sk, \cdot)$ perfectly with the aid of the CDH oracle $\mathcal{CDH}$.

Now we evaluate the Gap-DL advantage of $\mathcal{S}$. Let EXT denote the event that $g^{x^*} = X$ holds (that is, $\mathcal{H}'$ succeeds in extracting the exponent of $g^{x^*} = X$). If EXT occurs, then the solver $\mathcal{S}$ wins. So we have:

$$\Pr[\mathcal{S} \text{ wins}] \geqslant \Pr[\mathrm{EXT}].$$

Then, we do the following deformation;

$$\begin{aligned}
&\Pr[\mathcal{S} \text{ wins}] \\
\geqslant &\Pr[\mathcal{A} \text{ wins} \wedge \mathrm{EXT}] + \Pr[\neg(\mathcal{A} \text{ wins}) \wedge \mathrm{EXT}] \\
\geqslant &\Pr[\mathcal{A} \text{ wins} \wedge \mathrm{EXT}] \\
= &\Pr[\mathcal{A} \text{ wins}] - \Pr[\mathcal{A} \text{ wins} \wedge \neg \mathrm{EXT}].
\end{aligned}$$

Here $\mathcal{A}$ wins if and only if $\widehat{K}^* = Z = X^{a^*}$ holds. Therefore;

$$\Pr[\mathcal{S} \text{ wins}] \geqslant \Pr[\mathcal{A} \text{ wins}] - \Pr[X^{a^*} = Z \wedge g^{x^*} \neq X].$$

That means what we want.

$$\mathbf{Adv}^{\text{gap-dl}}_{\mathcal{S},\mathbf{Grp}}(k) \geqslant \mathbf{Adv}^{\text{ow-cca1}}_{\mathcal{A},\mathrm{EGKEM}}(k) - \mathbf{Adv}^{\text{kea}}_{\mathcal{H},\mathcal{H}',\mathbf{Grp}}(k) \quad \square$$

4.3 Discussion for Recommended Parameter Size

We write down parameter values of our concrete scheme. IFP denotes the Integer Factorization Problem and ECDLP denotes the Elliptic Curve Discrete Logarithm Problem.

Table 3 shows a set of parameter values.

When we use an e-mail address for identity data ID, 70 characters are available because in Table 3, there 564 bits remaining for identity data. That is, 70 byte. On condition that 1 character needs 1 byte, we can use 35 characters for identyty data of a candidate-participant. Also we can use 35 characters for identyty data of a guarantor.

図 3: Parameter Size (bit).

Security Paramter	λ	112
Equivalent Modulus Length for IFP	$\|N\|$	2048
Embeddable Information Length	$\|I\| = \|N\|/2 - \log_2(\|N\|) - 1$	1012
Equivalent Order Length for ECDLP	$\|p_0\| = 2\ \lambda$	224
Bit Length for Expression of a Point on a EC	$\|P\| = 2\|p_0\|$	448
Embeddable Length for Identity Data	$\|I\| - \|P\|$	564

Given (p_0, g, X) as an input;

Initial Setting
– Initialize its inner state
– $pk := (p_0, g, X)$, invoke $\mathcal{A}_1$ on pk

The First Phase : Answering $\mathcal{A}_1$'s Queries
– In the case that $\mathcal{A}_1$ queries $\mathcal{DEC}(sk, \cdot)$ for the answer for $\psi = h$;
• $Z \leftarrow \mathcal{CDH}(g, h, X)$, reply $\widehat{K} := Z$ to $\mathcal{A}_1$
– In the case that $\mathcal{A}_1$ returns the inner state st;
• Proceed to the Second Phase

The Second Phase : Extracting the Answer from $\mathcal{A}_2$'s Return
• $a^* \leftarrow \mathbb{Z}_q, \psi^* := h^* := g^{a^*}$
• Invoke $\mathcal{A}_2$ on (st, ψ^*)
– In the case that $\mathcal{A}_2$ returns $\widehat{K}^*$;
• Invoke $\mathcal{H}'$ on (g, h^*, st) : $x^* \leftarrow \mathcal{H}'(g, h^*, st)$
• Return x^*

図 2: A Gap-DL Problem Solver $\mathcal{S}$ for the Proof of Theorem 1.

5 Conclusions

We propose a decentralized identity management scheme by embedding identity information into a modulus of the public key of the RSA encryption. Then we provide the above decentralized identity management scheme in a concrete way by embedding a public key of an empolyed discrete-logarithm based encryption into a modulus of the RSA encryption.

6 Acknowledgements

The third autuhor was partially supported by the Invitation Programs for Foreign-based Researchers provided by NICT.

参考文献

[1] M. Bellare and A. Palacio. The Knowledge-of-Exponent Assumptions and 3-Round Zero-Knowledge Protocols. In *CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 273–289. Springer, 2004.

[2] 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.

[3] R. Canetti and R. R. Dakdouk. Extractable perfectly one-way functions. In *ICALP 2008*.

[4] R. R. Dakdouk. *Theory and Application of Extractable Functions*. PhD thesis, Yale University, New Haven, CT, USA, 2009.

[5] I. Damgård. Towards Practical Public Key Systems Secure against Chosen Ciphertext Attacks. In *CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 445–456. Springer, 1991.

[6] 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.

[7] 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.

[8] U. M. Maurer and S. Wolf. The Relationship between Breaking the Diffie-Hellman Protocol and Computing Discrete Logarithms. *SIAM J. Comput.*, 28(5):1689–1721, May 1999.

[9] 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.

[10] T. Sander and A. Ta-Shma. Auditable, Anonymous Electronic Cash. In *CRYPTO*, LNCS, pages 555–572. Springer, 1999.

[11] S. A. Vanstone and R. J. Zuccherato. Short RSA Keys and Their Generation. *J. Cryptology*, 8(2):101–114, 1995.

[12] 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.

[13] P. Zimmermann. Phil zimmermann's home page. `http://www.philzimmermann.com/EN/background/index.html`, 2014. [Online; accessed 25-Aug-2014].