A NEW TREE-STRUCTURE-SPECIFIED MULTISIGNATURE SCHEME FOR A DOCUMENT CIRCULATION SYSTEM

Masaki Inamura[1,2], Keiichi Iwamura[1], Ryu Watanabe[3], Makoto Nishikawa[2] and Toshiaki Tanaka[3]

[1]*Dept. of Electrical Engineering, Tokyo University of Science, 1–14–6 Kudankita, Chiyoda-ku, Tokyo, 102–0073 Japan*

[2]*Technology Development Center, KDDI R&D Laboratories Inc., 3–10–10 Iidabashi, Chiyoda-ku, Tokyo, 102–8460 Japan*

[3]*KDDI R&D Laboratories Inc., 2–1–15 Ohara, Fujimino-shi, Saitama, 356–8502 Japan*

Keywords: Intranet, Collaborative software, Digital signature, Multisignature, Gap-Diffie-Hellman group, Pairing.

Abstract: In this paper, the authors propose a new multisignature scheme with pairing-based cryptography, which can describe the tree structure of signers. In order to denote the relationship among a parent and its child signers, a dedicated middle key is generated on our scheme. In addition, we prove that our scheme is provably secure under the Gap-Diffie-Hellman assumption. Based on our proposal, we also implement the prototype of a document circulation system. In this system, a document is signed by members, who are divided into multiply layered groups. The browsing history can be confirmed by verifying the final signature on the document. The computational performance of the system is evaluated, and the result shows a good performance.

1 INTRODUCTION

Recently, a intranet becomes common technology and makes it possible for users to do their work efficiently. A information sharing system, document circulation system for work flow and schedule board system are examples (Digital Stamp Series). As to the document circulation system for work flow, it is important for managers to confirm who has already read a document. Therefore effective method is required for checking the state of the document. For this purpose, the multisignature scheme (Itakura and Nakamura, 1983) is one of the promising mechanisms. Under the multisignature scheme, because each member makes a signature to an objective document sequentially and all-members' signatures are aggregated to one signature called multisignature, a verifier can verify validity to all-members' signatures with verification of the only multisignature. Therefore the signing cost is minimized effectively. However, current multisignature schemes have a problem. The current multisignature schemes lack the function for describing of the relationship among the multisignature members. Each member, who forms a multisignature group has equal rights and duties for his / her signing. On the other hand, in a company, all workers are assigned various positions, and their rights and duties are also various. Therefore a new multisignature scheme, which can denote the hierarchical relationship among the members, is required.

In this paper, the authors propose a tree-structure-specified (TSS) multisignature scheme with a pairing-based cryptography called BLS signature (Boneh et al., 2001). Our contributions are described as follows:

- We propose a multisignature scheme, which can express order-specified structure of signers.
- We prove that our proposed multisignature scheme is provably secure under the Gap-Diffie-Hellman (GDH) assumption.
- We expand our proposed scheme as a TSS multisignature scheme.
- We apply this scheme to a document circulation system for intranets.

We also implement a prototype and evaluate its computational performance.

2 RELATED WORK

2.1 BLS Signature Scheme

Okamoto and Pointcheval defined a GDH class (Okamoto and Pointcheval, 2001). Consider a

362 Inamura M., Iwamura K., Watanabe R., Nishikawa M. and Tanaka T..
A NEW TREE-STRUCTURE-SPECIFIED MULTISIGNATURE SCHEME FOR A DOCUMENT CIRCULATION SYSTEM.
DOI: 10.5220/0003514903620369
In *Proceedings of the International Conference on Security and Cryptography* (SECRYPT-2011), pages 362-369
ISBN: 978-989-8425-71-3
Copyright © 2011 SCITEPRESS (Science and Technology Publications, Lda.)

multiplicative cyclic group $\mathbb{G}'$ with prime order p. They defined two problems for Diffie-Hellman (DH) as follows:

The Computational DH (CDH) Problem:
For $a,b \in \mathbb{Z}_p^*$ and $g \in \mathbb{G}'$, given (g, g^a, g^b), compute g^{ab}.

The Decisional DH (DDH) Problem:
For $a,b,c \in \mathbb{Z}_p^*$ and $g \in \mathbb{G}'$, given (g, g^a, g^b, g^c), decide whether $c \overset{?}{=} ab$.

As for the GDH class, it is defined as that the DDH problem is easy while the CDH problem is hard.

The first example of such groups was given by Joux and Nguyen (Joux and Nguyen, 2001; Joux and Nguyen, 2003), and Boneh, Lynn and Shacham showed that a new signature scheme based on the GDH class using pairing over elliptic curves could be structured (Boneh et al., 2001). Pairing is one of the functions on specified elliptic curves. Consider a group $\mathbb{G}$ on elliptic curves enabling pairing. Let e denote the symbol for pairing function. Pairing has the following characteristics:

- For $P_1, P_2, Q \in \mathbb{G}$,
  then $e(P_1 + P_2, Q) = e(P_1, Q)e(P_2, Q)$.
- For $P, Q_1, Q_2 \in \mathbb{G}$,
  then $e(P, Q_1 + Q_2) = e(P, Q_1)e(P, Q_2)$.
- For $a,b \in \mathbb{Z}_p^*$ and $P, Q \in \mathbb{G}$,
  then $e(aP, bQ) = e(bP, aQ) = e(abP, Q)$
  $= e(P, abQ) = e(P,Q)^{ab}$.

As a result, a signature scheme based on a group $\mathbb{G}$ can be structured as follows:

Key Generation: g is a generator of a group $\mathbb{G}$. The secret key of the signer is a random element $x \in \mathbb{Z}_p^*$ while his public key is $v = xg$.

Signing: H is a one-way hash function, which outputs a random element in the whole group $\mathbb{G}$, e.g. *MapToGroup* hashing onto $\mathbb{G}^*$ (Boneh et al., 2001). m is both a plain message and a signing target. The signer computes $h = H(m)$ and returns $\sigma = xh$.

Verification: When the verifier is given g, v, m and σ, he computes $h = H(m)$ and verifies $e(g,\sigma) \overset{?}{=} e(v,h)$.

When verifying, if a signer generates a signature correctly, a verifier can verify this signature with a result of pairing as follows:

$$e(g,\sigma) = e(g, xh) = e(g,h)^x.$$
$$e(v,h) = e(xg, h) = e(g,h)^x.$$

Therefore if $e(g,\sigma) = e(v,h)$, then σ has been generated correctly. Furthermore it has been proved that the above scheme is secure against chosen-message attacks (Boneh et al., 2001). In this paper, we call this signature scheme structured on the group $\mathbb{G}$ with pairing *the BLS signature scheme*.

After the concrete GDH signature scheme was proposed by Boneh et al., they, Boldyreva and Lin et al. proposed also a prototype of the multisignature scheme based on *the BLS signature scheme* (Boneh et al., 2003; Boldyreva, 2003; Lin et al., 2003). Consider a group $\mathbb{U} = \{u_1, \ldots, u_n\}$, which is a group of n signers, and a subgroup $\mathbb{L} = \{u_{i_1}, \ldots, u_{i_l}\} \subseteq \mathbb{U}$, which is a group of l members really participating in a multisignature. Let $\mathbb{J} = \{i_1, \ldots, i_l\}$ denote the set of indices of such members. Then proposers showed that we could structure the multisignature scheme based on *the BLS signature scheme* as follows:

Key Generation: g is a generator of a group $\mathbb{G}$. The secret key of the signer $u_i \in \mathbb{U}$ is a random element $x_i \in \mathbb{Z}_p^*$ while his public key is $v_i = x_i g$.

Signing: H is a one-way hash function, which outputs a random element in the whole group $\mathbb{G}$. m is both a plain message and a signing target. The signer $u_{i_\alpha} \in \mathbb{L}$ computes $h = H(m)$ and returns $\sigma_{i_\alpha} = x_{i_\alpha} h$.

Aggregation: The issuer of a multisignature finally collects all σ_{i_α} generated by u_{i_α}, computes $\sigma = \sum_{j\in\mathbb{J}} \sigma_j$ and returns $(m, \mathbb{L}, \sigma)$.

Verification: When the verifier is given $g, m, \mathbb{L}$ and σ, he collects all v_{i_α} by $\mathbb{L}$, computes $v = \sum_{j\in\mathbb{J}} v_j$ and $h = H(m)$, and verifies $e(g,\sigma) \overset{?}{=} e(v,h)$.

When verifying, if all signers generate a multisignature correctly, a verifier can verify this signature with a result of pairing as follows:

$$\begin{aligned} e(g,\sigma) &= e(g, \sum_{j\in\mathbb{J}} \sigma_j) = e(g, \sum_{j\in\mathbb{J}} x_j h) \\ &= e(g,h)^{\sum_{j\in\mathbb{J}} x_j}. \\ e(v,h) &= e(\sum_{j\in\mathbb{J}} v_j, h) = e(\sum_{j\in\mathbb{J}} x_j g, h) \\ &= e(g,h)^{\sum_{j\in\mathbb{J}} x_j}. \end{aligned}$$

Therefore if $e(g,\sigma) = e(v,h)$, then all σ_{i_α} have been generated correctly. Furthermore it has been proved that the above scheme is secure with reduction to the security of *the BLS signature scheme* (Boldyreva, 2003).

2.2 Multisignature Protocol Considering Hierarchical Relation

Several multisignature schemes considering hierarchical relation have been discussed.

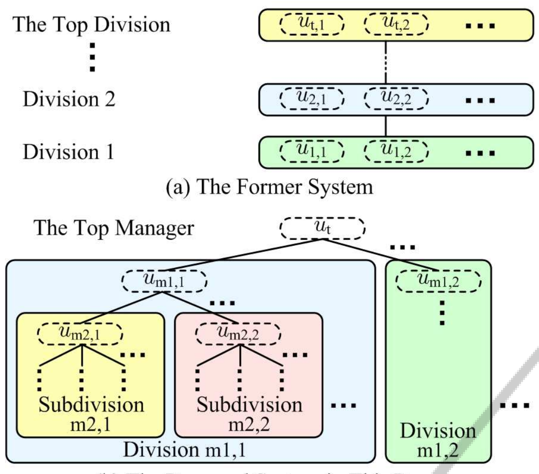

Figure 1: Difference of assuming system.

First an order-specified multisignature scheme, which can verify in which order each of signers signs a document, is proposed (Tada, 2003). However, this scheme cannot distinguish each division, which several signers belong to.

Lin et al. have proposed a concept of a hierarchical multisignature scheme, and so they discuss requirements for distinguishing each hierarchy (Lin et al., 2003). However, this scheme cannot be realized because multiplying the generator by the generator is executed without multiplying two elements in a GDH group.

We proposed the first realistic multisignature scheme, which constructed a hierarchical structure of signers (Inamura et al., 2010). In this scheme we introduced middle keys on purpose to express hierarchical relation among divisions, which each signer belonged to, and a verification protocol using these middle keys. Therefore they succeeded in verifying how hierarchy was. However, this scheme cannot verify complicated structures, e.g. a tree structure. Furthermore there is the problem that the number of middle keys increases in proportion to the depth of layers. Therefore we cannot adapt this scheme to assuming system, which is for a company constructed with accumulation of some divisions like a tree structure. We show a difference between two systems in Figure 1.

3 ORDER-SPECIFIED MULTISIGNATURE (PREPARATION)

In this section, for a preparation to propose the TSS multisignature, we propose a new order-specified multisignature scheme and security proof of the scheme.

3.1 Symbols, Preconditions and Requirements

3.1.1 Symbols

We define symbols used in our protocol as follows:

$\mathbb{G}$**:** A group on elliptic curves enabling pairing.

g**:** A generator in a group $\mathbb{G}$.

e**:** A pairing function.

u_o**:** A signer in our order-specified multisignature (o is the number of order).

x_o, v_o**:** A private key ($x_o \in \mathbb{Z}_p^*$) and a public key ($v_o \in \mathbb{G}$), which the signer u_o holds.

$\mathbb{L}_o$**:** Order information from u_1 to u_o.

V_o**:** A middle key for verifying an order-specified multisignature.

m**:** A plain message.

h**:** A one-way hash function onto $\mathbb{G}$ (e.g. *MapToGroup*).

σ_o**:** A signature to m based on *the BLS signature scheme*.

S_o**:** An order-specified multisignature from u_1 to u_o.

3.1.2 Preconditions

In our proposal, we assume preconditions as follows:

- Our order-specified multisignature scheme is operated with legal CA (Certification Authority).
- In the procedure of signing, all signers generate $\mathbb{L}_o$, V_o and σ_o rightfully.

3.1.3 Requirements

Komano et al. showed that we need to define security requirements to a multisignature scheme, which is different from those to a standard digital signature scheme (Komano et al., 2005; Komano et al., 2008). Therefore we define security requirements to our scheme are followings:

(1) Order-specified Legitimacy. Although a third party obtains all public information, the third party cannot add/delete signers' information, forge data. In addition, the third party cannot modify order in $\mathbb{L}_o$.

(2) Unforgeability. Although a third party obtains all public information, the third party cannot generate σ_o.

(3) Collusion-secure. A signer cannot collude with other signer for denial of Multisignature Participation.

(4) Impossibility of Deceptive Participation. A signer cannot let a third party participate in a multisignature without permission.

3.2 Protocol

3.2.1 Key Pair Generation

g is a generator of a group $\mathbb{G}$. The secret key of the signer u_i is a random element x_i (however, if $i \neq j$ then $x_i \neq x_j$) while his public key is $v_i = x_i g$.

When the signer u_i generates the key pair, u_i sends the public key to the CA. The CA generates random value $r \in \mathbb{Z}_p^*$ and sends it to u_i. u_i signs r using the secret key with *the BLS signature scheme* and sends the signature to the CA. The CA verifies the signature using the public key. If verification is succeeded, the CA registers the public key.

3.2.2 Signing & Aggregation

We show the procedure of signing and aggregation as follows:

S1. The signer u_1 in the first order computes the following:

$$\begin{aligned} h =& H(m). \\ \sigma_1 =& x_1 h. \\ S_1 =& \sigma_1. \\ \mathbb{L}_1 =& \{(0, u_1)\}. \\ V_1 =& v_1. \end{aligned}$$

This signer u_1 sends the signature set$(u_1, v_1, \sigma_1, S_1, \mathbb{L}_1, V_1)$ to a directly connected signer in the second order.

S2. The signer u_2 in the second order computes the following:

$$\begin{aligned} h =& H(m). \\ \sigma_2 =& x_2 h. \\ S_2 =& \sigma_2 + x_2\sigma_1. \\ \mathbb{L}_2 =& \mathbb{L}_1 + \{(u_1, u_2)\}. \\ V_2 =& v_2 + x_2 v_1. \end{aligned}$$

This signer u_2 sends the signature set$(u_2, v_2, \sigma_2, S_2, \mathbb{L}_2, V_2)$ to a directly connected signer in the third order.

S3. The signer u_o in the oth order computes the following:

$$\begin{aligned} h =& H(m). \\ \sigma_o =& x_o h. \\ S_o =& S_{o-1} + (x_o - 1)\sigma_{o-1} + \sigma_o. \\ \mathbb{L}_o =& \mathbb{L}_{o-1} + \{(u_{o-1}, u_o)\}. \\ V_o =& V_{o-1} + (x_o - 1)v_{o-1} + v_o. \end{aligned}$$

This signer u_o sends the signature set$(u_o, v_o, \sigma_o, S_o, \mathbb{L}_o, V_o)$ to a directly connected signer in the $(o+1)$th order.

S4. The signer $u_{o_{\text{last}}}$ in the last order computes the following:

$$\begin{aligned} h =& H(m). \\ \sigma_{o_{\text{last}}} =& x_{o_{\text{last}}} h. \\ S_{o_{\text{last}}} =& S_{o_{\text{last}}-1} + (x_{o_{\text{last}}} - 1)\sigma_{o_{\text{last}}-1} + \sigma_{o_{\text{last}}}. \\ \mathbb{L}_{o_{\text{last}}} =& \mathbb{L}_{o_{\text{last}}-1} + \{(u_{o_{\text{last}}-1}, u_{o_{\text{last}}})\}. \\ V_{o_{\text{last}}} =& V_{o_{\text{last}}-1} + (x_{o_{\text{last}}} - 1)v_{o_{\text{last}}-1} + v_{o_{\text{last}}}. \end{aligned}$$

This signer $u_{o_{\text{last}}}$ returns $(S_{o_{\text{last}}}, \mathbb{L}_{o_{\text{last}}}, V_{o_{\text{last}}})$ as the order-specified multisignature set.

3.2.3 Verification

When the verifier is given the order-specified multisignature set and has a mean to obtain all signers' public keys, he or she verifies the multisignature as follows:

V1. The verifier collects all v_o by $\mathbb{L}_{o_{\text{last}}}$.

V2. The verifier verifies the following:

$$e(g, V_{o_{\text{last}}}) \overset{?}{=} e(g, v_1)\Big(\prod_{i=2}^{o_{\text{last}}} e(v_{i-1}, v_i)\Big) \text{ by } \mathbb{L}_{o_{\text{last}}}.$$

V3. The verifier computes $h = H(m)$ and verifies $e(g, S_{o_{\text{last}}}) \overset{?}{=} e(V_{o_{\text{last}}}, h)$.

In the above scheme, the verifier can verify the legitimacy of a middle key $V_{o_{\text{last}}}$ generated by the signer $u_{o_{\text{last}}}$ using all public key $v_1, \ldots, v_{o_{\text{last}}}$. Therefore the verifier can realize that $V_{o_{\text{last}}}$ is correct, and then he can verify the Unforgeability of multisignature $\sigma_{o_{\text{last}}}$ in the same way as *the BLS signature scheme*.

3.3 Security Analysis

We defined security requirements to our scheme in section 3.1.3, so we discuss security analysis for those requirements in this section.

(1) Order-specified Legitimacy

Legitimacy of signing order depends on a numerical formula of the middle key $V_{o_{\text{last}}}$. Therefore if $V_{o_{\text{last}}}$ is secure, order of signers is correct.

We can prove security of order-specified legitimacy as follows:

Theorem 1.

Let $\mathbb{G}$ be a GDH group. Then $V_{o_{\text{last}}}$ is secure against forgery in the random oracle.

Proof of Theorem 1.

We can prove that $V_{o_{\text{last}}}$ is secure with reduction to the CDH problem. Let A be a poly-time adversary for the computing $V_{o_{\text{last}}}$. We will structure the adversary B for the CDH problem, such that
$\mathsf{Adv}^{\text{compute}}_{V_{o_{\text{last}}}}(A) = \mathsf{Adv}^{\text{ct-CDH}}_{\mathbb{G}}(B)$.
This means that if the CDH problem is hard then computing $V_{o_{\text{last}}}$ is secure.

It is obvious that A succeeds in forgery whenever B is successful. Therefore in this section, we proof the converse as follows:

Proof in case of $o_{\text{last}} = 1$ **:** $V_{o_{\text{last}}}$ is the same as $v_{o_{\text{last}}}$.

Proof in case of $o_{\text{last}} = 2$ **:** In this case, signers in the order structure are only two signers directly connected. B is having public keys v_1, v_2 and access to the random oracle. B will run A simulating the honest player. First B gives A the public keys v_1 and v_2 as the signers' key. x_1 and x_2 corresponding to the secret keys of v_1 and v_2 ought to exist. However, the adversary cannot realize x_1 and x_2 because of the discrete logarithm problem. B simulates A's random oracle using its own oracle. Whenever A asks the honest player to compute some middle keys, B forwards the query to its random oracle and return the reply back to A. At some point A outputs an attempted forgery V_2. Then B computes $x_1x_2g = V_2 - v_2$. It is easy to see that B simulates A for a valid experiment, runs in time comparable to running time of A and that it succeeds in forgery whenever A is successful.

Proof in case of $o_{\text{last}} = z(z \geq 3)$ **:** We assume that we have proved legitimacy when the number of order is $z-1$. B is having public keys v_{z-1}, v_z and access to the random oracle. B will run A simulating the honest player. First B gives A the public keys v_{z-1} and v_z as the signer's key in the $(z-1)$th order and in the last order. x_{z-1} and x_z corresponding to the secret keys of v_{z-1} and v_z ought to exist. However, the adversary cannot realize x_{z-1} and v_z because of the discrete logarithm problem. Then A outputs V_{z-1}. B simulates A's random oracle using its own oracle. Whenever A asks the honest player to compute some middle keys, B forwards the query to its random oracle and return the reply back to A. At some point A outputs an attempted forgery V_z. Then B computes $x_{z-1}x_zg = V_z - V_{z-1} + v_{z-1} - v_z$. It is easy to see that B simulates A for a valid experiment, runs in time comparable to the running time of A and that it succeeds in forgery whenever A is successful.

(QED)

(2) Unforgeability

An order-specified multisignature is $S_{o_{\text{last}}}$ in the end. Therefore if $S_{o_{\text{last}}}$ is secure, generated multisignature is correct.

We can prove security of unforgeability as follows:

Theorem 2.

Let $\mathbb{G}$ be a GDH group. Then $S_{o_{\text{last}}}$ is secure against forgery of the multisignature set in the random oracle.

Proof of Theorem 2.

We prove that $S_{o_{\text{last}}}$ is secure with reduction to *the BLS signature scheme*. Let A be a poly-time adversary for signing $S_{o_{\text{last}}}$. We will structure the adversary B for *the BLS signature scheme*, such that
$\mathsf{Adv}^{\text{multi-sign}}_{S_{o_{\text{last}}}}(A) = \mathsf{Adv}^{\text{sign}}_{\textit{the-BLS-signature}}(B)$.
This means that if *the BLS signature scheme* is secure (Boneh et al., 2001) then computing $S_{o_{\text{last}}}$ is secure.

If $V_{o_{\text{last}}}$ is open to the public for other multisignature, it is obvious that A succeeds in forgery whenever B is successful. Therefore in this section, we proof the converse as follows:
B is having a public key $v_{o_{\text{last}}}$ and access to the random oracle and the signing oracle. B will run A simulating the honest player. First B gives A the public keys $v_{o_{\text{last}}}$ as the signer's key in the last order. $x_{o_{\text{last}}}$ corresponding to the secret key of $v_{o_{\text{last}}}$ ought to exist. However, the adversary cannot realize $x_{o_{\text{last}}}$ because of the discrete logarithm problem. Then A outputs the set of $o_{\text{last}} - 1$ pairs of middle key and corresponding multiplicative secret key, which ought to be computed by signers $(v_1, x_1), \cdots, (v_{o_{\text{last}}-1}, x_{o_{\text{last}}-1})$, where $v_i = x_ig$. Let the above key pairs be child signers' directly connected with the root. B simulates A's random oracle using its own oracle. Whenever A asks the honest player to participate in the order-specified signature generation protocol on some message m, B forwards the query to its signing oracle and return the reply back to A. At some point A outputs an attempted forgery $S_{o_{\text{last}}}$. Then B computes
$x_{o_{\text{last}}}H(m)$
$= (x_{o_{\text{last}}-1} + 1)^{p-1}(S_{o_{\text{last}}} - (\sum_{i=2}^{o_{\text{last}}-1} x_{i-1}x_iH(m)))$.
It is easy to see that B simulates A for a valid experiment, runs in time comparable to running time of A and that it succeeds in forgery whenever A is successful. (QED)

(3) Collusion-secure

Connection between two signers (u_{o-1} and u_o) is indicated by the product of two secret keys of these signers ($x_{o-1}x_o$ included in $S_{o_{\text{last}}}$). If x'_{o-1} and x'_o of which $x'_{o-1}x'_o = x_{o-1}x_o$ consisted existed, u_{o-1} and u_o could deny participating in the multisignature with intrusion into other signers who might hold x'_{o-1} and x'_o.

As a countermeasure against the above, we consider that each secret key is a prime number in $\mathbb{Z}_p^*$. If any keys is generally a prime, so x'_{o-1} and x'_o of which $x'_{o-1}x'_o = x_{o-1}x_o$ consists do not exist.

(4) Impossibility of Deceptive Participation

A generated multisignature is following:

$$\begin{aligned} S_{o_{\text{last}}} =& x_1x_2h + \cdots + x_{o-1}x_oh + x_ox_{o+1}h + \dots \\ &+ x_{o_{\text{last}}-1}x_{o_{\text{last}}}h + x_{o_{\text{last}}}h. \end{aligned}$$

Then u_o attempts to let u'_o participate in the multisignature without permission in spite of nonparticipation in the multisignature. u_o obtains the public key v'_o of u'_o and calculates $v_d = v_o + v'_o$. If u_o could obtain x_d, which was obtained with logarithmic calculation of $v_d = x_dg$, u_o could let u'_o participate in the multisignature illegally with following calculation:

$$\begin{aligned} S'_{o_{\text{last}}} =& x_1x_2h + \cdots + x_{o-1}(x_d - x_o)h \\ &+ (x_d - x_o)x_{o+1}h + \cdots + x_{o_{\text{last}}-1}x_{o_{\text{last}}}h + x_{o_{\text{last}}}h \\ &(= x_1x_2h + \cdots + x_{o-1}x'_oh + x'_ox_{o+1}h + \dots \\ &\quad + x_{o_{\text{last}}-1}x_{o_{\text{last}}}h + x_{o_{\text{last}}}h). \end{aligned}$$

However, u_o cannot obtain x_d because of the discrete logarithm problem, so u_o cannot calculate the above.

4 BROWSING VERIFICATION SYSTEM (PROPOSITION)

4.1 Protocol

Our scheme in section 3 expresses order connection with repetition of the product of two secret keys, which directly connected signers hold. This expression is not limited to connect one to one, but enabled to expand into connecting one to many. Therefore we can propose a TSS multisignature with repetition of the connection between a parental signer and many signers in footings of children. Furthermore we can propose a browsing verification system of a document for circulating using this TSS multisignature.

In this section, we explain our new browsing verification system. As a sample case we consider our system on a binary tree structure with three layers in Figure 2.

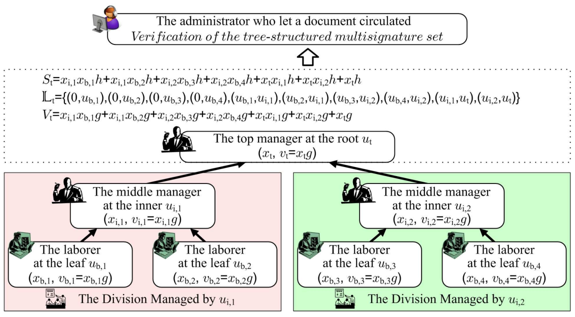
The administrator who let a document circulated
Verification of the tree-structured multisignature set
$S_t = x_{i,1}x_{b,1}h + x_{i,1}x_{b,2}h + x_{i,2}x_{b,3}h + x_{i,2}x_{b,4}h + x_tx_{i,1}h + x_tx_{i,2}h + x_th$
$\mathbb{L}_t = \{(0,u_{b,1}),(0,u_{b,2}),(0,u_{b,3}),(0,u_{b,4}),(u_{b,1},u_{i,1}),(u_{b,2},u_{i,1}),(u_{b,3},u_{i,2}),(u_{b,4},u_{i,2}),(u_{i,1},u_t),(u_{i,2},u_t)\}$
$V_t = x_{i,1}x_{b,1}g + x_{i,1}x_{b,2}g + x_{i,2}x_{b,3}g + x_{i,2}x_{b,4}g + x_tx_{i,1}g + x_tx_{i,2}g + x_tg$
The top manager at the root u_t $(x_t, v_t = x_tg)$
The middle manager at the inner $u_{i,1}$ $(x_{i,1}, v_{i,1} = x_{i,1}g)$
The middle manager at the inner $u_{i,2}$ $(x_{i,2}, v_{i,2} = x_{i,2}g)$
The laborer at the leaf $u_{b,1}$ $(x_{b,1}, v_{b,1} = x_{b,1}g)$
The laborer at the leaf $u_{b,2}$ $(x_{b,2}, v_{b,2} = x_{b,2}g)$
The laborer at the leaf $u_{b,3}$ $(x_{b,3}, v_{b,3} = x_{b,3}g)$
The laborer at the leaf $u_{b,4}$ $(x_{b,4}, v_{b,4} = x_{b,4}g)$
The Division Managed by $u_{i,1}$
The Division Managed by $u_{i,2}$

Figure 2: The outline of our system on a binary tree structure with three layers.

We redefine following symbols used in this binary tree structure with three layers:

$\mathbb{G}$**:** A group on elliptic curves enabling pairing.

g**:** A generator in a group $\mathbb{G}$.

e**:** A pairing function.

$u_{\mathbf{t}}$**:** A top manager at the root of the tree structure.

$u_{\mathbf{i},o_{\mathbf{i}}}$**:** Middle managers in the inner of the tree structure (o_{i} is the identity number in the inner of the tree structure).

$u_{\mathbf{b},o_{\mathbf{b}}}$**:** Laborers in the leaf of the tree structure (o_{b} is the identity number in the leaf of the tree structure).

$x_{\mathbf{t}}, v_{\mathbf{t}}$**:** A private key and a public key, which top manager u_{t} holds.

$x_{\mathbf{i},o_{\mathbf{i}}}, v_{\mathbf{i},o_{\mathbf{i}}}$**:** A private key and a public key, which a middle manager $u_{\text{i},o_{\text{i}}}$ holds.

$x_{\mathbf{b},o_{\mathbf{b}}}, v_{\mathbf{b},o_{\mathbf{b}}}$**:** A private key and a public key, which a laborer $u_{\text{b},o_{\text{b}}}$ holds.

$\mathbb{L}_{\mathbf{t}}$**:** Structured information from laborers to the top manager.

$\mathbb{L}_{\mathbf{i},o_{\mathbf{i}}}$**:** Structured information from laborers to a middle manager $u_{\text{i},o_{\text{i}}}$.

$\mathbb{L}_{\mathbf{b},o_{\mathbf{b}}}$**:** Information only a laborer $u_{\text{b},o_{\text{b}}}$.

$V_{\mathbf{t}}$**:** A middle key for verifying a TSS multisignature.

$V_{\mathbf{i},o_{\mathbf{i}}}, V_{\mathbf{b},o_{\mathbf{b}}}$**:** A sub-middle key for generating V_{t}.

m**:** A plain message in document for circulating.

h**:** A one-way hash function onto $\mathbb{G}$ (e.g. *MapToGroup*).

$\sigma_{\mathbf{i},o_{\mathbf{i}}}, \sigma_{\mathbf{b},o_{\mathbf{b}}}$**:** A signature to m of $u_{\text{i},o_{\text{i}}}$ or $u_{\text{b},o_{\text{b}}}$ based on *the BLS signature scheme*.

$S_{\mathbf{t}}$**:** A TSS multisignature.

$S_{\mathbf{i},o_{\mathbf{i}}}, S_{\mathbf{b},o_{\mathbf{b}}}$**:** A sub TSS multisignature for generating S.

The procedure for key pair generation is the same as the procedure in section 3.2. Therefore we omit the procedure for key pair generation.

4.1.1 Signing & Aggregation

We show the procedure of signing and aggregation as follows:

ST1. After the laborer $u_{\mathrm{b},1}$ browses the message m, this laborer computes the following:

$$\begin{aligned} h =& H(m). \\ \sigma_{\mathrm{b},1} =& x_{\mathrm{b},1}h. \\ S_{\mathrm{b},1} =& \sigma_{\mathrm{b},1}. \\ \mathbb{L}_{\mathrm{b},1} =& \{(0^{\dagger}, u_{\mathrm{b},1})\}. \\ & {}^{\dagger}0 \text{ means that } u_{\mathrm{b},1} \text{ is at the leaf.} \\ V_{\mathrm{b},1} =& v_{\mathrm{b},1}. \end{aligned}$$

This laborer $u_{\mathrm{b},1}$ sends the signature set$(u_{\mathrm{b},1}, v_{\mathrm{b},1}, \sigma_{\mathrm{b},1}, S_{\mathrm{b},1}, \mathbb{L}_{\mathrm{b},1}, V_{\mathrm{b},1})$ to a directly connected middle manager $u_{\mathrm{i},1}$. Other laborers $u_{\mathrm{b},o_{\mathrm{b}}}(2 \leq o_{\mathrm{b}} \leq 4)$ also execute the above procedure after they browse the message m and sends $(u_{\mathrm{b},o_{\mathrm{b}}}, v_{\mathrm{b},o_{\mathrm{b}}}, \sigma_{\mathrm{b},o_{\mathrm{b}}}, S_{\mathrm{b},o_{\mathrm{b}}}, \mathbb{L}_{\mathrm{b},o_{\mathrm{b}}}, V_{\mathrm{b},o_{\mathrm{b}}})$ to a middle manager (if $o_{\mathrm{b}} = 2$, then to $u_{\mathrm{i},1}$, else to $u_{\mathrm{i},2}$).

ST2. After the middle manager $u_{\mathrm{i},1}$ browses the message m, this middle manager computes the following:

$$\begin{aligned} h =& H(m). \\ \sigma_{\mathrm{i},1} =& x_{\mathrm{i},1}h. \\ S_{\mathrm{i},1} =& S_{\mathrm{b},1} + S_{\mathrm{b},2} \\ & + (x_{\mathrm{i},1} - 1)\sigma_{\mathrm{b},1} + (x_{\mathrm{i},1} - 1)\sigma_{\mathrm{b},2} + \sigma_{\mathrm{i},1}. \\ \mathbb{L}_{\mathrm{b},1} =& \mathbb{L}_{\mathrm{b},1} + \mathbb{L}_{\mathrm{b},2} + \{(u_{\mathrm{b},1}, u_{\mathrm{i},1}) + \{(u_{\mathrm{b},2}, u_{\mathrm{i},1})\}. \\ V_{\mathrm{i},1} =& V_{\mathrm{b},1} + V_{\mathrm{b},2} \\ & + (x_{\mathrm{i},1} - 1)v_{\mathrm{b},1} + (x_{\mathrm{i},1} - 1)v_{\mathrm{b},2} + v_{\mathrm{i},1}. \end{aligned}$$

This middle manager $u_{\mathrm{i},1}$ sends the signature set$(u_{\mathrm{i},1}, v_{\mathrm{i},1}, \sigma_{\mathrm{i},1}, S_{\mathrm{i},1}, \mathbb{L}_{\mathrm{i},1}, V_{\mathrm{i},1})$ to the top manager u_{t}. Another middle manager $u_{\mathrm{i},2}$ also executes the above procedure after they browse the message m and sends
$(u_{\mathrm{i},2}, v_{\mathrm{i},2}, \sigma_{\mathrm{i},2}, S_{\mathrm{i},2}, \mathbb{L}_{\mathrm{i},2}, V_{\mathrm{i},2})$ to u_{t}.

ST3. After the top manager u_{t} browses the message m, this top manager computes the following:

$$\begin{aligned} h =& H(m). \\ \sigma_{\mathrm{t}} =& x_{\mathrm{t}}h. \\ S_{\mathrm{t}} =& S_{\mathrm{i},1} + S_{\mathrm{i},2} + (x_{\mathrm{t}} - 1)\sigma_{\mathrm{i},1} + (x_{\mathrm{t}} - 1)\sigma_{\mathrm{i},2} + \sigma_{\mathrm{t}}. \\ \mathbb{L}_{\mathrm{t}} =& \mathbb{L}_{\mathrm{i},1} + \mathbb{L}_{\mathrm{i},2} + \{(u_{\mathrm{i},1}, u_{\mathrm{t}}) + \{(u_{\mathrm{i},2}, u_{\mathrm{t}})\}. \\ V_{\mathrm{t}} =& V_{\mathrm{i},1} + V_{\mathrm{i},2} + (x_{\mathrm{t}} - 1)v_{\mathrm{i},1} + (x_{\mathrm{t}} - 1)v_{\mathrm{i},2} + v_{\mathrm{t}}. \end{aligned}$$

The top manager u_{t} returns $(S_{\mathrm{t}}, \mathbb{L}_{\mathrm{t}}, V_{\mathrm{t}})$ as the TSS multisignature set.

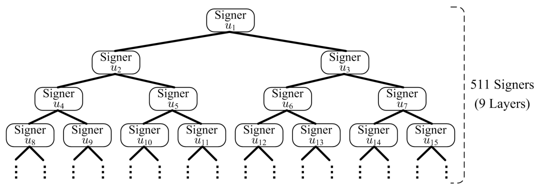

Figure 3: A binary tree structure in simulation program.

4.1.2 Verification

When the verifier, e.g. the administrator who let a document circulated, is given the TSS multisignature set and has a mean to obtain public keys of all members who have browsed a message, he or she verifies the multisignature as follows:

VT1. The verifier collects all $v_{\mathrm{t}}, v_{\mathrm{i},o_{\mathrm{i}}}, v_{\mathrm{b},o_{\mathrm{b}}}$ by $\mathbb{L}_{\mathrm{t}}$.

VT2. The verifier verifies the following:

$$\begin{aligned} e(g, V_{\mathrm{t}}) \overset{?}{=} & (\prod_{j=1}^{2} e(v_{\mathrm{b},j}, v_{\mathrm{i},1})) \cdot (\prod_{k=3}^{4} e(v_{\mathrm{b},k}, v_{\mathrm{i},2})) \\ & \cdot (\prod_{n=1}^{2} e(v_{\mathrm{i},n}, v_{\mathrm{t}})) \cdot e(g, v_t) \quad \text{by } \mathbb{L}_{\mathrm{t}}. \end{aligned}$$

VT3. The verifier computes $h = H(m)$ and verifies $e(g, \sigma_{\mathrm{t}}) \overset{?}{=} e(V_{\mathrm{t}}, h)$.

The above procedure has three following specifics:

- A numerical formula of the TSS multisignature is the same composition as that of the order-specified multisignature. Therefore we can explain security analysis of the TSS multisignature scheme with the same analysis in section 3.3.
- Our scheme generates the multisignature with repetition of the product of two secret keys, which directly connected managers or laborers hold. Therefore our scheme is suitable for any type of tree structure.
- Because of expansion of order-specified multisignature scheme, signing order is fixed. Therefore the verifier can know that a supervisor is sure to verify whether his or her directly connected subordinates browse a message or not.

As a result of the above procedure, the verifier can securely verify not only who browses this message but also in which order each of members circulates the document.

4.2 Performance Evaluation

We made the simulation program under the protocol in section 4.1 on a binary tree structure with 511 sign-

Table 1: Result of the running time on the simulation.

Signing at each signer.	Verification by verifier.
Max 0.125[sec]	3.838[sec]

ers cited in Figure 3 and measured the average running time for signing and verification on this program.

The specifications of the computer used for the simulation and of the security functions/parameters are the following:

Computer:

CPU: Intel Core i7 870.

Memory Size: 3.24GB.

OS: Microsoft Windows7 Ultimate (32bits).

Language: Microsoft Visual C++ 2008.

Security Functions/Parameters:

Pairing Function: Tate Pairing ($e(P,Q)$:The size of P is 20[Byte], The size of Q is 241[Byte]).

Hash Function: *MapToGroup* with SHA-256.

We show results of the running time for signing at the leaf, the inner and the root and for verification on the simulation program in Table 1.

Running time for signing at each signer is 0.125 seconds at the most. We have realized that the running time of addition is much less than that of scalar multiplication over an elliptic curve, and so multiplication dominates the running time of signing. Each signer computes two times multiplication, namely computation of a middle key and a middle signature. Therefore we suppose that signing time of each signer is almost constant.

On the other hand, the running time of verification is 3.838 seconds. When verification, the verifier needs to computes plural pairing function in proportion to the number of signers. Therefore the more the number of signers participating multisignature is, the more the calculation costs are. However, a result of this simulation shows that our system is realistic.

5 CONCLUSIONS

We proposed the order-specified multisignature scheme based on *the BLS signature scheme* and proved the security of our scheme with the random oracle. In addition, we expanded our order-specified multisignature scheme into the TSS multisignature scheme and adapted this scheme to the browsing verification system of a document for circulating.

REFERENCES

Digital Stamp Series. http://www.shachihata.co.jp/interweb/index.php.

Boldyreva, A. (2003). Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *Public Key Cryptography - PKC 2003, LNCS*, volume 2567, pages 31–46. Springer-Verlag.

Boneh, D., Gentry, C., Lynn, B., and Shacham, H. (2003). Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in Cryptology - EUROCRYPT 2003, LNCS*, volume 2656, pages 416–432. Springer-Verlag.

Boneh, D., Lynn, B., and Shacham, H. (2001). Short signatures from the weil pairing. In *Advances in Cryptology - ASIACRYPT 2001, LNCS*, volume 2248, pages 514–532. Springer-Verlag.

Inamura, M., Watanabe, R., and Tanaka, T. (2010). Proposal and evaluation of a hierarchical multisignature adapted to browsing verification of a document for circulating. *IEICE Trans. on Communications*, J93-B(10):1378–1387.

Itakura, K. and Nakamura, K. (1983). A public-key cryptosystem suitable for digital multisignatures. *NEC Research& Development*, 71:1–8.

Joux, A. and Nguyen, K. (2001). Separating decision diffie-hellman from diffie-hellman in cryptographic groups. *Cryptology ePrint Archive*, Report 2001/003.

Joux, A. and Nguyen, K. (2003). Separating decision diffie-hellman from computational diffie-hellman in cryptographic groups. *Springer J. of Cryptology*, 16(4):239–247.

Komano, Y., Ohta, K., Shimbo, A., and Kawamura, S. (2005). On the security of probabilistic multisignature schemes and their optimality. In *Cryptology in Malaysia - Mycrypt 2005, LNCS*, volume 3715, pages 132–150. Springer-Verlag.

Komano, Y., Ohta, K., Shimbo, A., and Kawamura, S. (2008). Provably secure multisignatures in formal security model and their optimality. *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, E91-A(1):107–118.

Lin, C. Y., Wu, T. C., and Zhang, F. (2003). A structured multisignature scheme from the gap diffie-hellman group. *Cryptology ePrint Archive*, Report 2003/090.

Okamoto, T. and Pointcheval, D. (2001). The gap-problems: A new class of problems for the security of cryptographic schemes. In *Public Key Cryptography - PKC 2001, LNCS*, volume 1992, pages 104–118. Springer-Verlag.

Tada, M. (2003). A secure multisignature scheme with signing order verifiability. *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, E86-A(1):73–88.