IEICE TRANS. FUNDAMENTALS, VOL.E84–A, NO.10 OCTOBER 2001

PAPER *Special Section on Information Theory and Its Applications*

A General Model of Multisignature Schemes with Message Flexibility, Order Flexibility, and Order Verifiability**

Shirow MITOMI[†*], ***Nonmember and*** **Atsuko MIYAJI**[†a)], ***Regular Member***

SUMMARY Multisignature scheme realizes that plural users generate the signature on a message, and that the signature is verified. Various studies on multisignature have been proposed [2], [6], [11], [15], [18]. They are classified into two types: RSA [13]-based multisignature [6], [11], and discrete logarithm problem (DLP) based multisignature [2], [15], [18], all of which assume that a message is fixed beforehand. In a sense, these schemes do not have a feature of message flexibility. Furthermore all schemes which satisfy with order verifiability designate order of signers beforehand [2], [18]. Therefore these protocols have a feature of order verifiability but not order flexibility. For a practical purpose of circulating messages soundly through Internet, a multisignature scheme with message flexibility, order flexibility and order verifiability should be required. However, unfortunately, all previous multisignature do not realize these features. In this paper, we propose a general model of multisignature schemes with flexibility and verifiability. We also present two practical schemes based on DLP based message recover signature [10] and RSA signature [6], respectively.

key words: *multisignature scheme, DLP-based signature, RSA signature, message recovery signature*

1. Introduction

In proportion as the spread of personal computers and network, messages like documents, data, software, etc., have been circulated through Internet. In such environment, an entity sends/forwards an original message to others, or sends a modified message to others. Through the process of circulation, a message has been improved or added a convenient feature one by one, and finally has been completed. However recently it has been a new problem for computer virus to be mixed into a message through the process of this circulation. Apparently it is an obstacle to circulate messages soundly through Internet. Another problem concerns the copyright: it is necessary to distinguish an original author from authors who modify an original message in a circulating message. This is why a multisignature scheme suitable for such an environment should be required.

Up to the present, various studies on multisignature have been proposed [2], [6], [11], [12], [15], [18]. They are classified into two types: RSA [13] based multisignature [6], [11], and discrete logarithm problem (DLP) based multisignature [2], [15], [18]. All schemes assume that a message is fixed beforehand since they suppose the following scenario: a message fixed beforehand is passed and signed one by one through members in an organization like a company. Therefore these schemes cannot handle the following situation: an original message is passed and modified by unspecified entities. Furthermore we want to guarantee such circulating message in the next point: who writes an original message, who modifies the message, to which the message is modified, and how order the message is modified. In previous multisignature schemes [2], [6], [11], [15], [18], signing from the first signer is obliged to start only if one of signers wants to modify a message: theses do not have a feature of *message flexibility*. Furthermore [6], [11], [15] have a feature of *order verifiability* neither. Order verifiability is first realized in [2], [18]. However they must designate order of signs beforehand. If we want to change order of signers, add a new signer, or exclude a signer, we are obliged to reset some data like public keys [2]: these have a feature of order verifiability but not *order flexibility*. Therefore previous schemes are not suitable for handling the above situation that a message circulates through unspecified entities.

In this paper, we propose a basic model of multisignature scheme that has the following three features:

Message flexibility: A message does not need to be fixed beforehand. Therefore each signer can modify an original message.

Order flexibility: Neither order of signers nor signers themselves need to be designated beforehand. Therefore we can easily change order of signers, add a new signer and exclude a signer.

Message and order verifiability: Each entity can verify who is an original author of a message, who modifies an original message and furthermore to which or how order a message is modified.

We also present two practical schemes based on the DLP based message recovery signature [10] and RSA signature [6]. Furthermore we discuss some typical attacks against our scheme like a ordinary forgery, swapping order of signers, excluding a signer. We denote the functions to break DLP, forge our scheme in ordinary assumption, that in swapping order of signers,

Manuscript received January 23, 2001.
Manuscript revised April 15, 2001.

[†]The authors are with Japan Advanced Institute of Science and Technology, Ishikawa-ken, 923-1292 Japan.
[*]Presently, with Fujitsu Co., Ltd.
a) E-mail: miyaji@jaist.ac.jp

[**]A preliminary version was presented at SCIS'2000 and ACISP'2000 [9].

and that in excluding a signer, by DLP, Forge, Swap, and Exclude, respectively. Then we prove the following theorems by using polynomial-time truth-table ($\leq_{tt}^{fp}$) reducibility of function:

(1) Forge $\equiv_{tt}^{fp}$ DLP, (2) Swap $\equiv_{tt}^{fp}$ DLP, and
(3) Exclude $\equiv_{tt}^{fp}$ DLP.

Furthermore we investigate a feature of *Robustness* in a multisignature scheme: a message cannot be recovered if the signature verification fails. Because unauthentic message might damage a receiver especially in case that a message circulate through unspecified entities. Therefore the following feature should be required:

Robustness: If the signature verification on a message fails, then prevent such an unauthentic message from damaging a receiver.

We also propose a general model of multisignature schemes with Robustness, *multisigncrypt*, which combines our multisignature with a function of encryption. Our multisigncrypt has a feature that a message cannot be recovered if the signature verification fails.

This paper is organized as follows. Section 2 summarizes a multisignature scheme [2] and discusses several drawbacks in case that a message circulate through unspecified entities. Section 3 investigates a model of multisignature with flexibility and verifiability. Section 4 presents two practical schemes concretely and discusses the performance. Section 5 discusses the security on our multisignature scheme. Section 6 presents our multisigncrypt scheme.

2. Previous Work

In this section, we summarize a previous multisignature scheme [2].

2.1 Previous Multisignature Scheme

We assume that n signers $I_1, I_2, ..., I_n$ generate a signature on a fixed message M according to order fixed beforehand.

Initialization: A trusted center generates a prime p, $g \in \mathbb{Z}_p^*$ with prime order q, and set a hash function $h()$. A signer I_i generates a random number $a_i \in \mathbb{Z}_q^*$ as I_i's secret key. Then I_i's public key is computed sequentially as follows: $y_1 = g^{a_1} \pmod p$, $y_i = (y_{i-1} \cdot g)^{a_i} \pmod p$. Then a public key of ordered group $(I_i, I_2, ..., I_i)$ is set to $y = y_n$.

Signature generation:
(1) Generation of r: Signer $I_1, ..., I_n$ generate r together as follows.

1. I_1 selects $k_1 \in \mathbb{Z}_q^*$ randomly and computes $r_1 = g^{k_1} \pmod p$. If $gcd(r_1, q) \neq 1$, then select new k_1 again.
2. For $i \in \{2, ..., n\}$; a signer I_{i-1} sends r_{i-1} to I_i. I_i selects $k_i \in \mathbb{Z}_q^*$ randomly and computes $r_i = r_{i-1}^{a_i} \cdot g^{k_i} \pmod p$. If $gcd(r_i, q) \neq 1$, then select new k_i again.
3. $r = r_i$.

(2) Generation of s: Signer $I_1, ..., I_n$ generate s together as follows.

1. I_1 computes $s_1 = a_1 + k_1 r \cdot h(r, M) \pmod q$.
2. For $i \in \{2, ..., n\}$; I_{i-1} sends s_{i-1} to I_i. I_i verifies that $g^{s_{i-1}} \stackrel{?}{=} y_{i-1} r_{i-1}^{r \cdot h(r,M)} \pmod p$, then computes $s_i = (s_{i-1} + 1)a_i + k_i r \cdot h(r, M) \pmod q$.
3. $s = s_i$.

(3) The multisignature on M by order $(I_1, ..., I_n)$ is given by (r, s).

Signature verification: A multisignature (r, s) on M is verified by checking $g^s \stackrel{?}{=} y \cdot r^{r \cdot h(r,M)} \pmod p$.

2.2 Drawbacks

In this section, we discuss the drawbacks of the previous scheme in the following situation: each entity sends an original message or a modified message to others. In such a situation, a multisignature scheme should satisfy the following conditions:

Message flexibility: A message does not need to be fixed beforehand. Therefore each signer can modify an original message.

Order flexibility: Neither order of signers nor signers themselves need to be designated beforehand. Therefore we can easily change order of signers, add a new signer and exclude a signer.

Message and order verifiability: Each entity can verify who is an original author of a message, who modifies an original message and furthermore to which or how order a message is modified.

The previous multisignature has the following drawbacks considering the above situation although it realizes order flexibility:

1. A message M should be fixed beforehand. This scheme does not allow any signer to generate a signature on his modified message.
2. A public key for multisignature should be determined by order of signers. Therefore after setting up a public key for multisignature, a signer can be neither added nor excluded. Even order of signers cannot be changed.
3. The signature generation phase runs two rounds through all signers.

3. Our Basic Multisignature Scheme

This section proposes a basic model of multisignature

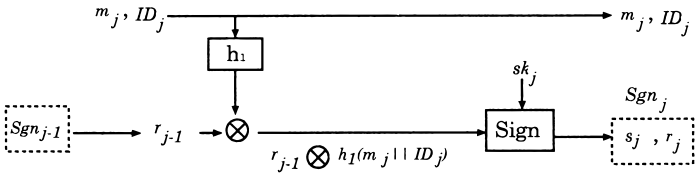


Fig. 1 I_j's signature generation.

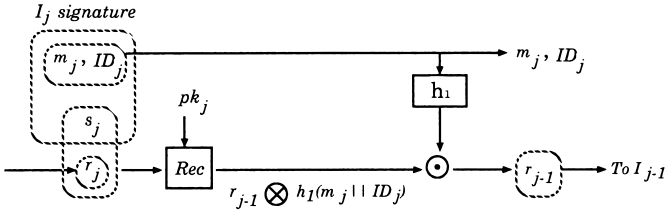


Fig. 2 I_j's signature verification step.

schemes with flexibility and verifiability for both message and order, which ensures the security such as the following situation.

Co-work environment: messages like documents, data, software, etc., have been developed independently, have been circulated among coworkers through Internet, have been improved or added a convenient feature one by one, and finally have been completed.

First we define the following notations. An original message M_1 is given by I_1. $M_{1,2,...,i}$ $(i > 2)$ denotes a message which is added some modification by the i-th signer I_i. The difference between $M_{1,2,...,i-1}$ and $M_{1,2,...,i}$, which means the modification by I_i, is defined as,

$$m_i = Diff(M_{1,2,...,i-1}, M_{1,2,...,i}).$$

We also define a function $Patch$ which recovers a message,

$$M_{1,2,...,i} = Patch(m_1, m_2, ..., m_i).$$

For the sake of convenience, we denote $m_1 = Patch(M_1)$. We use a signature scheme with a message recovery feature. The signature generation or message recovery function is denoted by $Sign(sk_i, m_i) = sgn_i$, or $Rec(pk_i, sgn_i) = m_i$, respectively, where sk_i is I_i's secret key and pk_i is I_i's public key. Let h_1 be a hash function, and ID_i be signer's identity information. We assume that the space of ID, $Space_{ID} = \{ID_i\}$ is sparse and discrete in $\{0,1\}^*$. We also use two operations $\otimes$ and $\odot$ in a group G

$$(A \otimes B) \odot B = A \ (\forall A, B \in G).$$

For example in case of $G = \mathbb{Z}_p$, $\otimes$ and $\odot$ mean modular multiplication and modular inversion, respectively. Then the signature generation and verification are done as follows. Figures 1 and 2 show the signature generation and verification, respectively.

Signature generation:

1. The first signer I_1 generates a signature on $h_1(m_1||ID_1)$ as follows,

   $$sgn_1 = Sign(sk_1, h_1(m_1||ID_1)) = (r_1, s_1),$$

   where a signature sgn_1 is divided into two parts, r_1 and s_1: r_1 is the next input to I_2's signature generation, which is recovered by I_2's signature verification. On the other hand, s_1 is the rest of sgn_1, which is sent to all signers as it is. Then send (ID_1, s_1, r_1, m_1) as a signature on m_1 to the next.
2. A signer I_j receives messages $m_1, m_2, ..., m_{j-1}$ from I_{j-1}. If $j > 2$, patch a message $M_{1,2,...,j-1}$ as follows,

   $$M_{1,2,...,j-1} = Patch(m_1, m_2, ..., m_{j-1}).$$

   I_j modifies $M_{1,2,...,j-1}$ to $M_{1,2,...,j-1,j}$, computes the modification m_j,

   $$m_j = Diff(M_{1,2,...,j-1}, M_{1,2,...,j}),$$

   and generates a signature on m_j by using r_{j-1} of I_j's signature,

   $$\begin{aligned} sgn_j &= Sign(sk_j, r_{j-1} \otimes h_1(m_j||ID_j)) \\ &= (r_j, s_j), \end{aligned}$$

   where sgn_j is divided into r_j and s_j in the same way as the above. Then I_j's signature on m_j is (r_j, s_j).
3. A multisignature on

   $$M_{1,2,...,i} = Patch(m_1, m_2, ..., m_i)$$

   by I_1, I_2, ..., I_{i-1} and I_i is given by (ID_1, s_1, m_1), (ID_2, s_2, m_2), $\cdots$, (ID_i, s_i, r_i, m_i).

Signature verification:

1. A verifier receives (ID_1, s_1, m_1), (ID_2, s_2, m_2), $\cdots$, (ID_i, s_i, r_i, m_i) from a signer I_i.

2. For $j = i, i-1, \cdots, 2$; compute

$$T_j = Rec(pk_j, (r_j, s_j)) = r_{j-1} \otimes h_1(m_j||ID_j),$$

$$r_{j-1} = T_j \odot h_1(m_j||ID_j).$$

Let $j = j-1$ and repeat step 2.

3. Finally compute

$$T_1 = Rec(PK_{p1}, (r_1, s_1)),$$

and verifies

$$T_1 \stackrel{?}{=} h_1(m_1||ID_1).$$

Our basic model satisfies the four features, message flexibility, order flexibility, message verifiability and order verifiability. Furthermore, we easily see that any message recovery signature can be applied to the above basic model. In the following sections, we present two schemes based on DLP and RSA.

4. A Concrete Multisignature Scheme

In this section, we give an example based on DLP. Another example based on RSA is described in Appendix.

4.1 DLP Based Multisignature Scheme

There are many variants of DLP based schemes in both types of message with appendix [3], [4], [17] and message recovery signature [1], [8], [10]. For the sake of convenience, here we uses the following basic scheme which is a message recovery signature scheme with DSA-signature equation [10]. Apparently any message recovery signature scheme such as the original NR-signature [10] can be applied to our multisignature scheme.

Basic scheme: The signer I generates a signature on a message m by using her/his secret key x of $y = g^x \pmod p$. First generate $k \in \mathbb{Z}_q$ randomly, and compute

$$R = g^k \pmod p,$$

$$r = R + m \pmod q,$$

$$s = (xr + 1)k^{-1} \pmod q.$$

Then the signature on m is (r, s), and m is recovered by computing $R' = g^{s^{-1}} y^{r \cdot s^{-1}} \pmod p$, and $m = r - R' \pmod q$.

We present one concrete multisignature scheme by using the above Basic scheme.

Initialization: An authenticated center generates a large prime p, $g \in \mathbb{Z}_p^*$ with prime order q. Two $\mathbb{Z}_p$-operations $\otimes$ and $\odot$ in section 3 are defined as multiplication and inverse in $\mathbb{Z}_p$, respectively. Each signer generates a pair of secret key $x_i \in \mathbb{Z}_q^*$ and a public key $y_i = g^{x_i} \pmod p$, and publish a public key y_i with his identity information ID_i.

Signature generation:

1. The first signer I_1 generates a signature on an original message m_1. First generate $k_1 \in \mathbb{Z}_q$ randomly, compute

$$R_1 = g^{k_1} \pmod p,$$

$$r_1 = R_1 + h_1(m_1||ID_1) \pmod q,$$

$$s_1 = (x_1 r_1 + 1)k_1^{-1} \pmod q,$$

where I_1's signature on m_1 is (r_1, s_1), and send (ID_1, s_1, r_1, m_1) to the next signer I_2.

2. A signer $I_j (j \geq 2)$ receives $M_{1,2,\cdots,j-1} = Patch(m_1, m_2, \cdots, m_{j-1})$, and modifies $M_{1,\cdots,j-1}$ to $M_{1,\cdots,j}$. Then I_j generates a signature on the difference $m_j = Diff(M_{1,\cdots,j-1}, M_{1,\cdots,j})$: generate $k_j \in \mathbb{Z}_q$ randomly, and compute

$$R_j = g^{k_j} \pmod p,$$

$$r_j = R_j + h_1(m_j||ID_j) \times r_{j-1} \pmod q,$$

$$s_j = (x_j r_j + 1)k_j^{-1} \pmod q,$$

where I_j's signature on m_j is (r_j, s_j).

3. A multisignature on

$$M_{1,2,\ldots,i} = Patch(m_1, m_2, ..., m_i)$$

by $I_1, \cdots, I_{i-1}$ and I_i is given by (ID_1, s_1, m_1), $\cdots$, $(ID_{i-1}, s_{i-1}, m_{i-1})$, (ID_i, s_i, r_i, m_i).

Signature verification:

1. A verifier receives (ID_1, s_1, m_1), $\cdots$, $(ID_{i-1}, s_{i-1}, m_{i-1})$ and (ID_i, s_i, r_i, m_i) from the signer I_i.
2. For $j = i, i-1, \cdots, 3, 2$; compute

$$R'_j = g^{s_j^{-1}} y_j^{r_j \cdot s_j^{-1}} \pmod p,$$

$$T_j = r_j - R'_j \pmod q, \text{ and}$$

$$r_{j-1} = T_j \cdot (h_1(m_j||ID_j))^{-1} \pmod q$$

by using I_j's public keys y_j. Let $j = j-1$ and repeat step 2.

3. Finally compute $R'_1 = g^{s_1^{-1}} y_1^{r_1 \cdot s_1^{-1}} \pmod p$, and $T_1 = r_1 - R'_1 \pmod q$, and verify $T_1 \stackrel{?}{=} h_1(m_1||ID_1) \pmod q$.

Our multisignature based on ElGamal-type signature has a feature that each signer has only one pair of a public key and a secret key.

Table 1 Performance of DLP-based multisignature schemes.

	Computation amount $\#M(1024)$		Signature size (bits)	$\#rounds$	Features
	I_i's signature generation	signature verification			
Our scheme	253	$302i$	$160(i+1)$	1	MF, MV, OF, OV
Primitive scheme	253	$291i$	$320i$	1	MF, MV, OF
Scheme [15]	242	$250+242i$	$160+1024i$	1	—
Scheme [2]	283	292	1,184	2	OV

MF: Message Flexibility, MV: Message Verifiability, OF: Order Flexibility, OV: Order Verifiability

4.2 Performance Evaluation

We evaluate our DLP-based multisignature scheme from a point of view of computation amount, the signature size and the number of rounds, where the signature size means that the final multisignature by $I_1, \cdots, I_i$, and the number of rounds means how many times the process to generate the signature runs among all signers. There has not been proposed a multisignature with message flexibility, order flexibility and order verifiability. One primitive scheme with message flexibility is a simple chain of signature: each signer makes a signature on his own modification and sends it together with the previous signer's signature. Apparently it does not satisfy order verifiability. We also compare our schemes with the primitive scheme, which is based on DSA-signature. For a simple discussion, we assume the following conditions: (1) a primitive arithmetic of binary methods [7] is used for computation of exponentiation; (2) we denote the number of signers, the computation time for one n-bit modular multiplication and that for one n-bit modular inversion by i, $M(n)$ and $I(n)$, respectively; (3) we assume that $M(n) = (\frac{m}{n})^2 M(m)$ and that $I(n) = 10M(n)$; (4) two primes p and q are set to 1024 and 160 bits respectively, in DLP-based signature schemes.

DLP based-multisignature schemes are mainly classified into two types, one-round scheme [15] and two-round scheme in Sect. 2. Generally, the signature verification phase in two-round scheme is more simple than one-round scheme. However the signature generation phase in two-round scheme, which runs twice through all signers, is rather complicated. Here we compare our scheme with the primitive scheme, one-round scheme [15] and two-round scheme [2]. Table 1 shows performance of 4 schemes. From Table 1, we see that the computation amount for signature verification increases only a little bit, and that the signature size is even reduced, compared with the same one-round multisignature. Therefore our protocol can realize four features with message flexibility, order flexibility, message verifiability, and order verifiability only with negligible additional computation amount in signature generation.

5. Security Consideration

In this section, we discuss the security relation between our DLP based multisignature scheme and DLP. Here we aim at such a situation that there exist attackers among signers, and that they try to forge not only a message but also signer's order. Therefore we assume that all signers except for an honest signer I_n collude in attacks: attackers use all secret keys $x_j (j \neq n)$, random numbers k_j, public information like public keys, all messages $m_1, \cdots, m_n \in \mathbb{Z}$, all identity information $ID_1, \cdots, ID_n \in \mathbb{Z}$, and so their hash values, $h_1(m_1||ID_1) = H_1, \cdots, h_1(m_n||ID_n) = H_n \in \mathbb{Z}$, and valid partial signatures. By using these informations, attackers try to forge I_i's signatures. For simplicity, we denote the sequence $x_1, x_2, ..., x_n$ by $x_{[1,n]}$ and the sequence $x_1, x_2, ..., x_{i-1}, x_{i+1}, ..., x_n$ by $x_{[1,n,i]}$, where $1 \leq i \leq n$. We also denote $x_1, x_2, ..., x_n \in \mathbb{Z}_q$ by $x_{[1,n]} \in \mathbb{Z}_q$.

Generally speaking, two types of attacks in the security proof are required: the passive attack as the first step and the active attack as the next step [5]. Our scheme gives a general model of multisignature schemes with message flexibility, order flexibility, and their verifiability for the first time. Therefore as the first step we discuss the precise security model in the passive attack. In our security proof, we use the polynomial-time truth-table ($\leq^{fp}_{k-tt}$) reducibility of the function version [14]. In $\leq^{fp}_{k-tt}$ only k non-adaptive queries to an oracle are allowed. Here we simply write $\leq^{fp}_{tt}$ because we do not have to stress the number of queries.

5.1 Functions

First we define some functions.

Definition 1: $\mathtt{DLP}(X, g, p, q)$ is the function that on input two primes p, q with $q|(p-1)$, $X, g \in \mathbb{Z}^*_p$ outputs $a \in \mathbb{Z}_q$ such that $X = g^a \pmod p$ if such $a \in \mathbb{Z}_q$ exists.

We define the function `Forge` that forges I_n's valid signature (r_n, s_n) on $m_{[1,n]}$ in order $I_{[1,n]}$ by using available public information, a signature on $m_{[1,n-1]}$ by $I_{[1,n-1]}$ and available secret data like $x_{[1,n-1]}$ and $k_{[1,n-1]}$ for attackers $I_{[1,n-1]}$.

Definition 2: $\mathtt{Forge}(y_n,\ g,\ p,\ q,\ H_{[1,n]},\ x_{[1,n-1]},$

$s_{[1,n-1]}$, r_{n-1}, k_n) is the function that on input two primes p, q with $q|(p-1)$, y_n, $g \in \mathbb{Z}_p^*$, $s_{[1,n-1]}$, r_{n-1}, $x_{[1,n-1]}$, k_n, $H_{[1,n]} \in \mathbb{Z}_q^*$, outputs $(r_n, s_n) \in \mathbb{Z}_q^* \times \mathbb{Z}_q^*$ such that $R_n = g^{k_n}$, $R_n = y_n^{r_n/s_n} g^{1/s_n}$, and $r_n = R_n + H_n r_{n-1}$, for $j = n-1, ..., 3, 2$: $R_j = g^{s_j^{-1}} y_j^{r_j \cdot s_j^{-1}} \pmod p$, $T_j = r_j - R_j \pmod q$, and $r_{j-1} = T_j \cdot H_j^{-1} \pmod q$, and $R_1 = g^{s_1^{-1}} y_1^{r_1 \cdot s_1^{-1}} \pmod p$, $T_1 = r_1 - R_1 \pmod q$, and $T_1 = H_1 \pmod q$, where $y_i = g^{x_i} \pmod p (i = 1, \cdots, n-1)$ if such $(r_n, s_n) \in \mathbb{Z}_q^* \times \mathbb{Z}_q^*$ exists.

Next we define the function `Exclude` that forges I_n's valid signature (s'_n, k_n) on $m_{[1,n,n-1]}$ in order $I_{[1,n,n-1]}$ by using available public information, a signature on $m_{[1,n]}$ by $I_{[1,n]}$ and available secret data $x_{[1,n-1]}$ and $k_{[1,n-1]}$ for attackers $I_{[1,n-1]}$.

Definition 3: `Exclude`(y_n, g, p, q, $H_{[1,n]}$, $x_{[1,n-1]}$, $s_{[1,n]}$, r_{n-2}, r_n) is the function that on input two primes p, q with $q|p-1$, g, $y_n \in \mathbb{Z}_p^*$, $x_{[1,n-1]}$, r_n, r_{n-2}, $s_{[1,n]}$, $H_{[1,n]} \in \mathbb{Z}_q^*$, outputs $(s'_n, k_n) \in \mathbb{Z}_q^* \times \mathbb{Z}_q^*$ such that $R'_n = g^{k_n} \pmod p$, $r'_n = H_n \times r_{n-2} + R'_n \pmod q$, $R'_n = g^{s'^{-1}_n} y_n^{r'_n \cdot s'^{-1}_n} \pmod p$, for $j = n-2, \cdots, 2$: $R_j = g^{s_j^{-1}} y_j^{r_j \cdot s_j^{-1}} \pmod p$, $T_j = r_j - R_j \pmod q$, and $r_{j-1} = T_j \cdot H_j^{-1} \pmod q$, and $R_1 = g^{s_1^{-1}} y_1^{r_1 \cdot s_1^{-1}} \pmod p$, $T_1 = r_1 - R_1 \pmod q$, and $T_1 = H_1 \pmod q$, where $y_i = g^{x_i} \pmod p (i = 1, \cdots, n-1)$ if such $(s'_n, k_n) \in \mathbb{Z}_q^* \times \mathbb{Z}_q^*$ exists.

Next we define the function `Swap` that forges valid multisignature on $m_{[1,n-2]}$, m_n, m_{n-1} in order $I_{[1,n-2]}$, I_n, I_{n-1} by using available public information, a valid multisignature $(r_n, s_{[1,n]})$ on $m_{[1,n]}$ by $I_{[1,n]}$ and available secret data $x_{[1,n-1]}$ and $k_{[1,n-1]}$ of attackers $I_{[1,n-1]}$. From the assumption that $I_{[1,n-1]}$ are attackers, the function `Swap` that forges I_n's signature (r_n, s_n) on $m_{[1,n-2]}$, m_n, m_{n-1} in order $I_{[1,n-2]}$, I_n, I_{n-1} for a valid signature $(r_n, s_{[1,n]})$ on $m_{[1,n]}$ by $I_{[1,n]}$ is just the same as the function that computes `Exclude` and adds attacker I_{n-1}'s signature on $m_{[1,n-2]}$, m_n, m_{n-1} in order $I_{[1,n-2]}$, I_n, I_{n-1}. Oppositely, the function `Exclude` is just the same as the function that for a valid signature $(r_n, s_{[1,n]})$ on $m_{[1,n]}$ by $I_{[1,n]}$, computes `Swap` and outputs only I_n's multisignature (r_n, s_n). Therefore the following theorem holds.

Theorem 1: `Swap` $\equiv_{tt}^{fp}$ `Exclude`.

Next we investigate the function `Exchange` that forges valid multisignature on $m_{[1,n]}$ in order $I_{[1,n-2]}$, I_n, I_{n-1} by using available public information, a valid multisignature $(r_n, s_{[1,n]})$ on $m_{[1,n]}$ by $I_{[1,n]}$ and available secret data $x_{[1,n-1]}$ and $k_{[1,n-1]}$ of attackers $I_{[1,n-1]}$. `Swap` changes both order of signers and order of messages, but `Exchange` changes only order of signers. From the assumption that $I_{[1,n-1]}$ are attackers, the function `Exchange` that forges I_n's signature (r'_n, s'_n) on $m_{[1,n]}$ in order $I_{[1,n-2]}$, I_n, I_{n-1} for a valid signature $(r_n, s_{[1,n]})$ on $m_{[1,n]}$ by $I_{[1,n]}$ is just the same as the function that computes `Forge` in the case of which forges I_n's valid signature (r_n, s_n) on $m_{[1,n-1]}$ in order $I_{[1,n-2]}$, I_n by using available public information, a signature on $m_{[1,n-2]}$ by $I_{[1,n-2]}$ and available secret data like $x_{[1,n-2]}$ and $k_{[1,n-2]}$ for attackers $I_{[1,n-2]}$, and adds attacker I_{n-1}'s signature on $m_{[1,n]}$ in order $I_{[1,n-2]}$, I_n, I_{n-1}. Therefore as for `Exchange` it is enough to investigate the security of `Forge`. More strictly the following theorem holds.

Theorem 2: `Exchange` $\leq_{tt}^{fp}$ `Forge`.

Finally we define the function `Attack` in order to discuss the relation between the basic scheme and the multisignature scheme. The function `Attack` that forges I_n's valid signature (r, s) on m in the basic scheme by using information of multisignatures such as available public information, a signature on $m_{[1,n]}$ by $I_{[1,n]}$ and available secret data like $x_{[1,n-1]}$ and $k_{[1,n-1]}$ of attackers $I_{[1,n-1]}$.

Definition 4: `Attack`(y_n, g, p, q, $H_{[1,n]}$, $x_{[1,n-1]}$, $s_{[1,n]}$, r_n, r_{n-1}, m) is the function that on input two primes p, q with $q|(p-1)$, $y_n, g \in \mathbb{Z}_p^*$, $s_{[1,n]}$, r_n, $x_{[1,n-1]}$, $H_{[1,n]}, m \in \mathbb{Z}_q^*$, outputs $(r, s) \in \mathbb{Z}_q^* \times \mathbb{Z}_q^*$ such that $R = y_n^{r/s} g^{1/s}$, and $m = R - r$ if such $(r, s) \in \mathbb{Z}_q^* \times \mathbb{Z}_q^*$ exists.

For the sake of the following proof, we define the function `SIGN` that generates a valid signature, including all partial signatures, $(r_{[1,n]}, s_{[1,n]})$ on messages $m_{[1,n]}$ by signers $I_{[1,n]}$ by using all secret data $x_{[1,n]}$ and $k_{[1,n]}$ of signers $I_{[1,n]}$. This function means just the signature generation function. Apparently it is easy to compute `SIGN`.

Definition 5: `SIGN`$(g, p, q, x_{[1,n]}, k_{[1,n]}, H_{[1,n]})$ is the function that on input two primes p, q with $q|(p-1)$, $g \in \mathbb{Z}_p^*$, $x_{[1,n]}, k_{[1,n]}$, $H_{[1,n]} \in \mathbb{Z}_q^*$, outputs $r_{[1,n]}, s_{[1,n]} \in \mathbb{Z}_q^*$ such that for $j = n, ..., 3, 2$: $R_j = g^{s_j^{-1}} y_j^{r_j \cdot s_j^{-1}} \pmod p$, $T_j = r_j - R_j \pmod q$ and $r_{j-1} = T_j \cdot H_j^{-1} \pmod q$; $R_1 = g^{s_1^{-1}} y_1^{r_1 \cdot s_1^{-1}} \pmod p$, $T_1 = r_1 - R_1 \pmod q$, $T_1 = H_1 \pmod q$, where $y_i = g^{x_i} \pmod p (i = 1, \cdots, n)$ if such $r_{[1,n]}, s_{[1,n]} \in \mathbb{Z}_q$ exists.

5.2 Reduction among Functions

Here we show our results. First we set functions ψ_i to give the i-th element, $\psi_i(a_{[1,n]}) = a_i (i \leq n)$.

Theorem 3: `Forge` $\equiv_{tt}^{fp}$ `DLP`

Proof: First we show that `Forge` $\leq_{tt}^{fp}$ `DLP`. For inputs $(y_n, g, p, q, H_{[1,n]}, x_{[1,n-1]}, s_{[1,n-1]}, r_{n-1}, k_n)$ of `Forge`, set $R_n = g^{k_n} \pmod p$, $r_n = r_{n-1} \cdot H_n + R_n \pmod q$. Then

$$\begin{aligned}&\texttt{Forge}(y_n, g, p, q, H_{[1,n]}, x_{[1,n-1]}, s_{[1,n-1]}, r_{n-1})\\ &\quad = (r_n, (\texttt{DLP}(y_n, g, p, q) r_n + 1) k_n^{-1} \pmod q)\\ &\quad = (r_n, s_n).\end{aligned}$$

Next we show that $\texttt{DLP} \leq_{tt}^{fp} \texttt{Forge}$. For input ($y_n$, g, p, q) of DLP, fix $k_{[1,n]} \in \mathbb{Z}_q^*$, $H_{[1,n]} \in \mathbb{Z}$, $x_{[1,n-1]} \in \mathbb{Z}_q^*$, and set

$$\begin{aligned}&(r_{[1,n-1]}, s_{[1,n-1]})\\ &\quad = \texttt{SIGN}(g, p, q, x_{[1,n-1]}, k_{[1,n-1]}, H_{[1,n-1]}),\end{aligned}$$

which is computed in time polynomial from the definition. Then

$$\begin{aligned}&\texttt{DLP}(y_n, g, p, q)\\ &\quad = (\psi_2(\texttt{Forge}(y_n, g, p, q, H_{[1,n]}, x_{[1,n-1]},\\ &\qquad s_{[1,n-1]}, r_{n-1}, k_n)) \cdot k_n - 1) r_n^{-1},\end{aligned}$$

where

$$\begin{aligned}r_n \;=\; &\psi_1(\texttt{Forge}(y_n, g, p, q, H_{[1,n]}, x_{[1,n-1]}, s_{[1,n-1]},\\ &r_{n-1}, k_n)).\end{aligned}$$

Therefore we get $\texttt{DLP} \equiv_{tt}^{fp} \texttt{Forge}$. □

Theorem 4: $\texttt{Exclude} \equiv_{tt}^{fp} \texttt{DLP}$

Proof: First we show that $\texttt{Exclude} \leq_{tt}^{fp} \texttt{DLP}$. For inputs ($y_n$, g, p, q, $H_{[1,n]}$, $x_{[1,n-1]}$, $s_{[1,n]}$, r_{n-2}, r_n) of Exclude, fix $k_n \in \mathbb{Z}_q$, and set $R_n = g^{k_n} \pmod p$, and $r'_n = r_{n-2} \cdot H_n + R_n \pmod q$. Then

$$\begin{aligned}&\texttt{Exclude}(y_n, g, p, q, H_{[1,n]}, x_{[1,n-1]}, s_{[1,n]}, r_{n-2}, r_n)\\ &\quad = ((\texttt{DLP}(y_n, g, p, q) r'_n + 1) k_n^{-1} \pmod q, k_n).\end{aligned}$$

Next we show that $\texttt{DLP} \leq_{tt}^{fp} \texttt{Exclude}$. For inputs ($y_n$, g, p, q) of DLP, fix $k_{[1,n-1]} \in \mathbb{Z}_q^*$, $H_{[1,n-1]} \in \mathbb{Z}$, $x_{[1,n-1]} \in \mathbb{Z}_q^*$, $s_n, r_n \in \mathbb{Z}_q^*$ and set

$$\begin{aligned}&(r_{[1,n-1]}, s_{[1,n-1]})\\ &\quad = \texttt{SIGN}(g, p, q, x_{[1,n-1]}, k_{[1,n-1]}, H_{[1,n-1]}),\end{aligned}$$

which is computed in time polynomial from the definition. Furthermore set $R_n = g^{s_n^{-1}} y_n^{r_n \cdot s_n^{-1}} \pmod p$ and $H_n = (r_n - R_n)/r_{n-1}$. Then

$$\texttt{DLP}(y_n, g, p, q) = (s'_n \cdot k_n - 1) \cdot {r'}_n^{-1},$$

where

$$\begin{aligned}s'_n \;=\; &\psi_1(\texttt{Exclude}(y_n, g, p, q, H_{[1,n]}, x_{[1,n-1]},\\ &s_{[1,n]}, r_{n-2}, r_n),\end{aligned}$$

$$\begin{aligned}k_n \;=\; &\psi_2(\texttt{Exclude}(y_n, g, p, q, H_{[1,n]}, x_{[1,n-1]},\\ &s_{[1,n]}, r_{n-2}, r_n)),\end{aligned}$$

$$R'_n = g^{k_n} \pmod p, \text{ and}$$

$$r'_n = r_{n-2} \cdot H_1 + R'_n \pmod q.$$

Therefore we get $\texttt{DLP} \equiv_{tt}^{fp} \texttt{Exclude}$. □

Theorem 5: $\texttt{Attack} \leq_{tt}^{fp} \texttt{Forge}$

Proof: For input (y_n, g, p, q, $H_{[1,n]}$, $x_{[1,n-1]}$, $s_{[1,n]}$, r_n, r_{n-1}, m) of Attack, fix $k_n \in \mathbb{Z}_q^*$, and set $H'_i = H_i (i = 1, \cdots, n-1)$, and $H'_n = m/r_{n-1} \pmod p$. Then

$$\begin{aligned}&\texttt{Attack}(y_n, g, p, q, H_{[1,n]}, x_{[1,n-1]}, s_{[1,n]}, r_n, r_{n-1}, m)\\ &\quad = \texttt{Forge}(y_n, g, p, q, H'_{[1,n]}, x_{[1,n-1]}, s_{[1,n-1]},\\ &\qquad r_{n-1}, k_n).\end{aligned}$$

Therefore we get $\texttt{Attack} \leq_{tt}^{fp} \texttt{Forge}$. □

From Theorem 5, we see that Forge includes such an attack against the basic scheme by using available information on multisignatures.

6. Further Discussion

We discuss how to add the following feature to our multisignature scheme.

Robustness: If the signature verification fails, then prevent such an unauthentic message from damaging a receiver.

We realize robustness by combining our multisignature with an encryption function. So we call it *multisigncrypt*. Multisigncrypt has a feature that a message cannot be recovered if the signature verification fails, in addition to message flexibility, order flexibility, and order verifiability. Therefore a multisigncrypt can prevent computer virus mixed into a message from damaging a receiver since unauthentic message can not be recovered.

6.1 Multisigncrypt Scheme

For simplicity, we present the multisigncrypt scheme by using our basic multisignature scheme.

Initialization: A center publishes two hash functions h_1 and h_2, and an encryption and the decryption function, $E(K_i, m_i)$ and $D(K_i, C_i)$, in addition to initialization in basic multisignature scheme, where h_2 is used for computing a session key K_i for E and D, and C_i is a cipher text. Figures 3 and 4 show the signature generation and verification, respectively.

Signature generation:

1. The first signer I_1 computes

   $$sgn_1 = sign(sk_1, h_1(m_1||ID_1)) = (r_1, s_1),$$

   where sgn_1 is divided into two parts of r_1 and s_1 in the same way as Sect. 3, generates a session key K_1,

   $$K_1 = h_2(h_1(m_1||ID_1)),$$

   and encrypts $m_1||ID_1$ by an encryption function E,

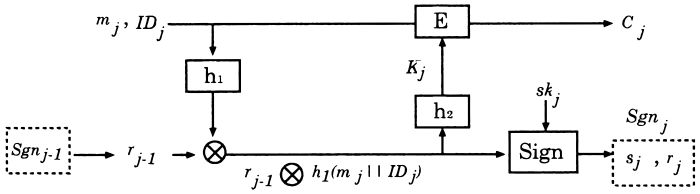


Fig. 3 I_j's signature generation.

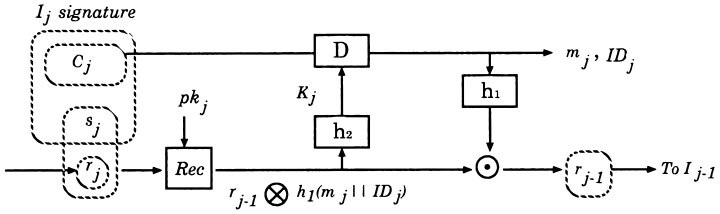


Fig. 4 I_j's signature verification step.

$$C_1 = E(K_1, m_1||ID_1),$$

and sends (ID_1, s_1, r_1, C_1) to the next signer I_2.

2. A signer I_j verifies the signature on $m_1, \cdots, m_{j-1}$ from I_{j-1} according to the verification step in the next paragraph, and modifies $M_{1,\cdots,j-1} = Patch(m_1, \cdots, m_{j-1})$ to $M_{1,\cdots,j}$. Then I_j generates a signature on the difference $m_j = Diff(M_{1,\cdots,j-1}, M_{1,\cdots,j-1,j})$: compute

$$\begin{aligned} sgn_j &= Sign(sk_j, r_{j-1} \otimes h_1(m_j||ID_j)) \\ &= (r_j, s_j), \end{aligned}$$

$$K_j = h_2(r_{j-1} \otimes h_1(m_j||ID_j)),$$

and encrypts $m_j||ID_j$ by using the session key K_j,

$$C_j = E(K_j, m_j||ID_j).$$

3. A multisignature on

$$M_{1,2,\ldots,i} = Patch(m_1, m_2, ..., m_i)$$

by I_1, $\cdots$, I_i is given by (ID_1, s_1, C_1), (ID_2, s_2, C_2), $\cdots$, (ID_i, s_i, r_i, C_i).

Signature verification:

1. The verifier receives (ID_1, s_1, C_1), $\cdots$, (ID_{i-1}, s_{i-1}, r_{i-1}, C_{i-1}), (ID_i, s_i, r_i, C_i) from the signer I_i.
2. For $j = i, ..., 3, 2$: compute

$$T_j = Rec(pk_j, (s_j, r_j)), \text{ and } K_j = h_2(T_j),$$

and decrypts m_j and ID_j by

$$m'_j||ID'_j = D(K_j, C_j).$$

If $ID'_j \stackrel{?}{=} ID_j$ holds, then accept the signature and recover r_{j-1},

$$r_{j-1} = T_j \odot h_1(m'_j||ID'_j).$$

Set $j = j - 1$ and repeat step 2.

3. Compute

$$T_1 = Rec(pk_1, (s_1, r_1)) \text{ and } K_1 = h_2(T_1),$$

and decrypt m_1 and ID_1 by

$$m'_1||ID'_1 = D(K_1, C_1).$$

If $h_1(m'_1||ID'_1) \stackrel{?}{=} T_1$ holds, then accept the signature and finally patch all messages,

$$M_{1,\cdots,i} = Patch(m_1, \cdots, m_i).$$

In both cases of DLP- and RSA-based multisignature schemes, we can also add the feature of Robustness in the same way as the above.

7. Conclusion

In this paper, we have proposed a new multisignature scheme suitable for circulating messages through Internet. Our multisignature scheme realizes the four features, Message flexibility, Order flexibility, Message verifiability and Order verifiability, maintaining both signature size and computation amount in signature generation/verification low: the computation amount for the signature verification increases only a little bit, and the signature size is even reduced compared with one round previous multisignature scheme. We have also proposed the multisigncrypt scheme, which realizes Robustness in addition to Message flexibility, Order flexibility, Message verifiability, and Order verifiability. Furthermore, we have proved the following equivalences between our DLP-based multisignature and DLP in some typical attacks by using the reducibility of functions.

1. Forge$\equiv_{tt}^{fp}$DLP,
2. Swap$\equiv_{tt}^{fp}$DLP,
3. Exclude$\equiv_{tt}^{fp}$DLP.

References

[1] M. Abe and T. Okamoto, "A signature scheme with message recovery as secure as discrete logarithm," Advances in Cryptology—Proc. ASIACRYPT'99, Lecture Notes in Computer Science, vol.1716, pp.378–389, Springer-Verlag, 1999.

[2] M. Burmester, Y. Desmedt, H. Doi, M. Mambo, E. Okamoto, M. Tada, and Y. Yoshifuji, "A structured ElGamal-Type multisignature scheme," Advances in Cryptology—Proc. PKC'2000, Lecture Notes in Computer Science, pp.466–482, Springer-Verlag, 2000.

[3] "Specification for a digital signature standard," National Institute for Standards and Technology, Federal Information Standard Publication XX, draft, 1991.

[4] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," IEEE Trans. Inf. Theory, vol.IT-31, no.4, pp.469–472, 1985.

[5] S. Goldwasser, S. Micali, and R.L. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," SIAM J. Computing, vol.17, no.2, pp.281–308, 1988.

[6] K. Itakura and K. Nakamura, "A public-key cryptosystem suitable for digital multisignatures," NEC J. Res. Dev.71, Oct. 1983.

[7] D.E. Knuth, The art of computer programming, vol.2, Seminumerical Algorithms, 2nd ed., Addison-Wesley, Reading, Mass., 1981.

[8] A. Miyaji, "Another countermeasure to forgeries over message recovery signature," IEICE Trans. Fundamentals, vol.E80-A, no.11, pp.2192–2200, Nov. 1997.

[9] S. Mitomi and A. Miyaji, "A multisignature scheme with message flexibility, order flexibility and order verifiability," Information Security and Privacy—Proc. ACISP 2000, Lecture Notes in Computer Science, vol.1841, pp.298–312, Springer-Verlag, 2000.

[10] K. Nyberg and R.A. Rueppel, "Message recovery for signature schemes based on the discrete logarithm problem," Designs Codes and Cryptography, vol.7, pp.61–81, 1996.

[11] T. Okamoto, "A digital multisignature scheme using bijective public-key cryptosystems," ACM Trans. Computer Syst., vol.6, no.8, pp.432–441, 1988.

[12] T. Okamoto and K. Ohta, "A digital multisignature scheme based on the Fiat-Shamir scheme," Advances in Cryptology—Proc. ASIACRYPT'91, Lecture Notes in Computer Science, vol.739, pp.139–148, Springer-Verlag, 1993.

[13] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," Commun. ACM, vol.21, no.2, pp.120–126, 1978.

[14] K. Sakurai and H. Shizuya "Relationships among the computational powers of breaking discrete log cryptosystem," Advanced in Cryptology—Proc. Eurocrypt'95, Lecture Notes in Computer Science, vol.921, pp.341–355, Springer-Verlag, 1995. (J. Cryptology, vol.11, pp.29–43, 1998.)

[15] A. Shimbo, "Multisignature schemes based on the Elgamal scheme," The 1994 Symposium on Cryptography and Information Security, vol.SCIS94-2C, Jan. 1994.

[16] R.D. Silverman, "A cost-based security analysis of symmetric and asymmetric key length," CryptoBytes in RSA Laboratories, vol.13, 1999.

[17] C.P. Schnorr, "Efficient signature generation by smart cards," J. Cryptology, vol.4, pp.161–174, 1991.

[18] T. Saito, "A multiple signature scheme enabling a specified Signer's order," The 1997 Symposium on Cryptography and Information Security, vol.SCIS97-33A, Jan. 1994.

Appendix: Another Concrete Multisignature Scheme

A.1 RSA-Based Multisignature Scheme

Here we present our multisignature scheme based on RSA multisignature [6].

Initialization: An authenticated center publishes small primes in addition to $\{1\}$, $\{r_l\} = \{1, 2, 3, 5, \cdots\}$. A signer I_i with identity information ID_i generates two large primes p_i and q_i secretly, and computes public keys $n_{i,l}$ and $e_{i,l} \in \mathbb{Z}^*_{n_{i,l}}$ in such a way that

$$n_{i,l} = p_i q_i r_l, (l \geq 1)$$

$$L_{i,l} = \begin{cases} LCM((p_i - 1), (q_i - 1)) \\ \quad (l = 1) \\ LCM((p_i - 1), (q_i - 1), (r_l - 1)) \\ \quad (l \geq 2) \end{cases}$$

$$e_{i,l} d_{i,l} = 1 \pmod{L_{i,l}},$$

by using $\{r_l\}$. For the sake of convenience, we denote $n_i = p_i q_i (= n_{i,1})$ and $e_i = e_{i,1}$. Signer I_i publishes all his public keys $n_{i,l}$, $e_{i,l}$ and r_l like Table A·1. Let $n_{min} = \min\{n_i\}_i$, and h_1 be a hash function to $\mathbb{Z}_{n_{min}}$, where $\min\{n_i\}_i$ means the minimum integer of $\{n_i\}_i$.

In RSA-based multisignature, both operations in $\mathbb{Z}_{n_{i,l}}$ $\otimes$ and $\odot$ are set to $\oplus$ (EOR), and I_i's signature sgn_i is just the next input to I_{i+1}'s signature generation: sgn_i is not divided into two parts.

Signature generation:

1. The first signer I_1 generates a signature on an original message m_1: select a minimum number n_{1,l_1} such that $n_{1,l_1} > h_1(m_1||ID_1)$ and compute $sgn_1 = (h_1(m_1||ID_1))^{d_{1,l_1}} \pmod{n_{1,l_1}}$. Then send (ID_1, m_1, l_1, sgn_1) as a signature on m_i to the next.
2. A signer I_j receives $m_1, m_2, ..., m_{i-1}$ from I_{j-1}. If $j > 2$, patch the message $M_{1,2,...,j-1} = Patch(m_1, m_2, ..., m_{j-1})$, modify it to $M_{1,2,...,j}$. Then I_j generates a signature on $m_j = Diff(M_{1,2,...,j-1}, M_{1,2,...,j-1,j})$: select a minimum number n_{j,l_j} such that $n_{j,l_j} > sgn_{j-1} \oplus h_1(m_j||ID_j)$, and compute $T = sgn_{j-1} \oplus h_1(m_j||ID_j)$, and $sgn_j = T^{d_{j,l_j}} \pmod{n_{j,l_j}}$.
3. A multisignature on

$$M_{1,2,...,i} = Patch(m_1, m_2, ..., m_i)$$

Table A·1 I_i's pairs of secret key and public key.

l	1	2	$\cdots$ $\cdots$
r_l	r_1	r_2	$\cdots$ $\cdots$
public keys	$(n_{i,1}, e_{i,1})$	$(n_{i,2}, e_{i,2})$	$\cdots$ $\cdots$
secret keys	$d_{i,1}$	$d_{i,2}$	$\cdots$ $\cdots$

Table A· 2 Performance of RSA based signatures.

	Computation amount $\#M(1024)$		Signature size (bits)	$\#rounds$	Features
	I_i's signature generation	signature verification			
Our scheme	1536	$9i$	$1024+10i$	1	MF, MV, OF, OV
Primitive scheme	1536	$9i$	$1024i$	1	MF, MV, OF

MF: Message Flexibility, MV: Message Verifiability, OF: Order Flexibility, OV: Order Verifiability

by $I_1, \cdots, I_{i-1}$ and I_i is given by (ID_1, l_1, m_1), (ID_2, l_2, m_2), $\cdots$, and (ID_i, l_i, m_i, sgn_i).

Signature verification:

1. The verifier receives (ID_1, l_1, m_1), (ID_2, l_2, m_2), $\cdots$, (ID_i, l_i, m_i, sgn_i) from a signer I_i.
2. For $j = i, i-1, ..., 2$; compute

$$T' = (sgn_j)^{e_{j,l_j}} \pmod{n_{j,l_j}},$$

$$sgn_{j-1} = h_1(m_j||ID_j) \oplus T'$$

by using I_j's public key (n_{j,l_j}, e_{j,l_j}). Let $j = j-1$ and repeat step 2.
3. Compute $T' = sgn_1^{e_{1,l_1}} \pmod{n_{1,l_1}}$ by using I_1's public key (n_{1,l_1}, e_{1,l_1}), and check $T' \stackrel{?}{=} h_1(m_1||ID_1)$.

In our multisignature scheme, order of signers does not have to be fixed beforehand. Therefore even if all public keys $\{n_i\}$ are set to be the same size, such a case as $n_{i+1} < n_i$ may happen. This is why we need an additional set of $\{r_l\}$. The number of signers in a series of multisignatures might be limited according as $\{r_l\}$. However from a practical point of view, it does not seem to cause a serious problem considering the number of signers for a series of multisignatures.

Our multisignature based on RSA has the following features: (1) The size of multisignature keeps low even if the number of signers increases, compared with DLP based scheme. (2) It is necessary for each signer to have plural pairs of secret and public key.

A.2 Performance Evaluation

We evaluate our RSA-based multisignature scheme from a point of view of computation amount, and the signature size, where the signature size means that the final multisignature by $I_1, \cdots, I_i$. There has not been proposed a multisignature with message flexibility, order flexibility and order verifiability. One primitive scheme with message flexibility is a simple chain of signature: each signer makes a signature on his own modification and sends it together with the previous signer's signature. Apparently it does not satisfy order verifiability. We also compare our schemes with the primitive scheme. For a simple discussion, we assume the following conditions: (1) a primitive arithmetic of binary methods [7] is used for computation of exponentiation; (2) we denote the number of signers and the computation time for one n-bit modular multiplication by i and $M(n)$, respectively, where $M(n) = (\frac{m}{n})^2 M(m)$; (3) two primes p_j and q_j are set to 512 bits, and r_l is less than 10 bits in RSA-based signature schemes.

Here we compare our RSA-based multisignature scheme with the primitive scheme. Table A· 2 shows performance of two schemes. From Table A· 2, we see that our protocol can realize four features, message flexibility, order flexibility, message verifiability and order verifiability, with the same computation amount as the primitive scheme. Note that the signature size is even reduced.

A.3 Security Consideration

In this section, we investigate the security relation between our RSA based multisignature scheme and RSA[†] in the same point as Sect. 5. Here we discuss some of attack models since all results hold in almost the same way as Sect. 5. For simplicity, we denote the sequence $n_{1,1}, n_{1,2}, \cdots, n_{1,l}, n_{2,1}, \cdots, n_{j,l}$, by $n_{[1,j],[1,l]}$, and $d_{1,1} \in \mathbb{Z}_{n_{1,1}}, \cdots, d_{j,l} \in \mathbb{Z}_{n_{j,l}}$ by $d_{[1,j],[1,l]} \in \mathbb{Z}_{n_{[1,j],[1,l]}}$ in addition to notations defined in Sect. 5.

A.4 Functions

First we define some functions.

Definition 6: $\mathtt{RSA}(n, m, e)$ is the function that on input an integer $n \in \mathbb{Z}$, $m, e \in \mathbb{Z}_n$, outputs $s \in \mathbb{Z}_n$ such that $s^e = m \pmod{n}$ if such $s \in \mathbb{Z}_n$ exists.

We define the function `Forge` that forges I_j's valid signature (sgn_j, l_j) on $m_{[1,j]}$ in order $I_{[1,j]}$ by using available public information, a signature on $m_{[1,j-1]}$ by $I_{[1,j-1]}$ and available secret data such as $p_{[1,j-1]}$, $q_{[1,j-1]}$ and $d_{[1,j-1],[1,l]}$ of attackers $I_{[1,j-1]}$.

Definition 7: $\mathtt{Forge\text{-}RSA}(n_{j,[1,l]}, r_{[1,l]}, e_{j,[1,l]}, H_{[1,j]}, p_{[1,j-1]}, q_{[1,j-1]}, d_{[1,j-1],[1,l]}, sgn_{[1,j-1]}, l_{[1,j-1]})$ is the function that on input $n_{j,[1,l]} \in \mathbb{Z}$, $e_{j,[1,l]} \in \mathbb{Z}_{n_{j,[1,l]}}$, primes $p_{[1,j-1]}$, $q_{[1,j-1]}$, $r_{[1,l]}$, and $d_{[1,j-1],[1,l]} \in$

[†]The original RSA signature [13] sets n to a product of two primes, that is $n = pq$. For simplicity, here we use a generalized model of RSA in which the number of prime factors of n is not limited to 2, and the recent result of [16] is included. Apparently the original RSA problem is reduced to the generalized RSA problem.

$\mathbb{Z}_{n_{[1,j-1],[1,l]}}$, $H_{[1,j]} \in \mathbb{Z}_{n_{min}}$, $l_{[1,j-1]} \in \{1,\cdots,l\}$, $sgn_1 \in \mathbb{Z}_{n_{1,l_1}}, \cdots, sgn_{j-1} \in \mathbb{Z}_{n_{j-1,l_{j-1}}}$, outputs $(sgn_j, l_j) \in \mathbb{Z}_{n_{j,l_j}} \times \{1,\cdots,l\}$ such that for i=$j,\ldots,3,2$: T_i=$sgn_i{}^{e_{i,l_i}} \pmod{n_{i,l_i}}$; $sgn_{i-1} = H_i \oplus T_i$, T_1=$sgn_1{}^{e_{1,l_1}} \pmod{n_{1,l_1}}$, $T_1 = H_1$ where $n_{i,t} = p_i q_i r_t (1 \le i \le j-1, 1 \le t \le l)$, and $n_{min} = \min\{n_{[1,j],1}\}_j$, if such (sgn_j, l_j) exists, and otherwise outputs $\perp$.

Next we define the function Exclude that forges I_j's valid signature (sgn'_j, l'_j) on $m_{[1,j,j-1]}$ in order $I_{[1,j,j-1]}$ by using available public information, a signature on $m_{[1,j]}$ by $I_{[1,j]}$ and available secret data such as $p_{[1,j-1]}$, $q_{[1,j-1]}$ and $d_{[1,j-1],[1,l]}$ of attackers $I_{[1,j-1]}$.

Definition 8: Exclude-RSA$(n_{j,[1,l]}, r_{[1,l]}, e_{j,[1,l]}, H_{[1,j]}, p_{[1,j-1]}, q_{[1,j-1]}, d_{[1,j-1],[1,l]}, sgn_{[1,j]}, l_{j-2}, l_j)$ is the function that on input $n_{j,[1,l]} \in \mathbb{Z}$, $e_{j,[1,l]} \in \mathbb{Z}_{n_{j,[1,l]}}$, primes $p_{[1,j-1]}$, $q_{[1,j-1]}$, $r_{[1,l]}$, and $d_{[1,j-1],[1,l]} \in \mathbb{Z}_{n_{[1,j-1],[1,l]}}$, $H_{[1,j]} \in \mathbb{Z}_{n_{min}}$, $sgn_1 \in \mathbb{Z}_{n_{1,l_1}}, \cdots, sgn_j \in \mathbb{Z}_{n_{j,l_j}}$, $l_{j-2}, l_j \in \{1,\cdots,l\}$, outputs $(sgn'_j, l'_j) \in \mathbb{Z}_{n_{j,l'_j}} \times \{1,\cdots,l\}$ such that $T_j = sgn'_j{}^{e_{j,l'_j}} \pmod{n_{j,l'_j}}$; $sgn_{j-2} = H_j \oplus T_j$, for $i = j-2,...,3,2$: $T_i = sgn_i{}^{e_{i,l_i}} \pmod{n_{i,l_i}}$; $sgn_{i-1} = H_i \oplus T_i$, $T_1 = sgn_1{}^{e_{1,l_1}} \pmod{n_{1,l_1}}$, $T_1 = H_1$ if such (sgn'_j, l'_j) exists, and otherwise outputs $\perp$.

As for the function Swap-RSA that forges valid multisignature on $m_{[1,j-2]}$, m_j, m_{j-1} in order $I_{[1,j-2]}$, I_j, I_{j-1} by using available public information, a valid multisignature on $m_{[1,j]}$ by $I_{[1,j]}$ and available secret data of attackers $I_{[1,j-1]}$, we can easily get the following result in the same way as Theorem 2.

Theorem 6: Swap-RSA $\equiv_{tt}^{fp}$ Exclude-RSA.

For the sake of the following proof, we define the function SIGN-RSA that generates a valid signature, including all partial signatures, $(sign_{[1,j]}, l_{[1,j]})$ on messages $m_{[1,j]}$ by signers $I_{[1,j]}$ by using all secret data such as $p_{[1,j]}$ and $q_{[1,j]}$ of signers $I_{[1,j]}$. This function means just the signature generation function. Apparently it is easy to compute SIGN-RSA.

Definition 9: SIGN-RSA$(n_{[1,j],[1,l]}, H_{[1,j]}, d_{[1,j],[1,l]}, e_{[1,j],[1,l]})$ is the function that on input $n_{[1,j],[1,l]} \in \mathbb{Z}$, $e_{[1,j],[1,l]} \in \mathbb{Z}_{n_{[1,j],[1,l]}}$, the corresponding RSA-keys $d_{[1,j],[1,l]}$, and $H_{[1,n]} \in \mathbb{Z}_{n_{min}}$, outputs $(l_{[1,j]}, sgn_1, \cdots, sgn_j) \in \{1,l\} \times \mathbb{Z}_{n_{1,l_1}} \times \cdots \times \mathbb{Z}_{n_{j,l_j}}$ such that for $i = j,...,3,2$: $T_i = sgn_i{}^{e_{i,l_i}} \pmod{n_{i,l_i}}$; $sgn_{i-1} = H_i \oplus T_i$, $T_1 = sgn_1{}^{e_{1,l_1}} \pmod{n_{1,l_1}}$, $T_1 = H_1$, where $n_{min} = \min\{n_{i,1}\}_i$ if such $(sgn_{[1,j]}, l_{[1,j]})$ exists, and otherwise outputs $\perp$.

Here we show our results.

Theorem 7: Forge-RSA $\equiv_{tt}^{fp}$ RSA

Proof: First we show that Forge-RSA $\le_{tt}^{fp}$RSA. For inputs $(n_{j,[1,l]}, r_{[1,l]}, e_{j,[1,l]}, H_{[1,j]}, p_{[1,j-1]}, q_{[1,j-1]}, d_{[1,j-1],[1,l]}, sgn_{[1,j-1]}, l_{[1,j-1]})$ of Forge-RSA, set $T = sgn_{j-1} \oplus H_j$, and set the minimum integer l_j such that $n_{j,l_j} > T$. If such l_j does not exist, then output $\perp$. Then by using (n_{j,l_j}, e_{j,l_j}),

$$\begin{aligned}&\texttt{Forge-RSA}(n_{j,[1,l]}, r_{[1,l]}, e_{j,[1,l]}, H_{[1,j]}, p_{[1,j-1]},\\ &\quad q_{[1,j-1]}, d_{[1,j-1],[1,l]}, sgn_{[1,j-1]}, l_{[1,j-1]})\\ &= (\texttt{RSA}(n_{j,l_j}, T, e_{j,l_j}), l_j).\end{aligned}$$

Next we show that RSA $\le_{tt}^{fp}$ Forge-RSA. For inputs (n, m, e) of RSA, set $r_1 = 1$, $n_j = n$, $e_{j,1} = e$, fix $e_{j,[2,l]} \in \mathbb{Z}_n$, primes $p_{[1,j-1]}$, $q_{[1,j-1]}$, and $r_{[2,l]}$, and computes key pairs of $(n_{[1,j-1],[1,l]}, d_{[1,j-1],[1,l]}, e_{[1,j-1],[1,l]})$ from $p_{[1,j-1]}$, $q_{[1,j-1]}$ and $r_{[1,l]}$. Then set $n_{min} = \{n_{i,1}\}_i$, fix $H_{[1,j-1]} \in \mathbb{Z}_{n_{min}}$, and set

$$\begin{aligned}&(sgn_{[1,j-1]}, l_{[1,j-1]})\\ &\quad = \texttt{SIGN-RSA}(n_{[1,j-1],[1,l]}, H_{[1,j-1]}, d_{[1,j-1],[1,l]},\\ &\qquad e_{[1,j-1],[1,l]}),\end{aligned}$$

and $H_j = m \oplus sgn_{j-1}$. Then

$$\begin{aligned}&\texttt{RSA}(n, m, e)\\ &\quad = \psi_1(\texttt{Forge}-\texttt{RSA}(n_{j,[1,l]}, r_{[1,l]}, e_{j,[1,l]}, H_{[1,j]},\\ &\qquad p_{[1,j-1]}, q_{[1,j-1]}, d_{[1,j-1],[1,l]},\\ &\qquad sgn_{[1,j-1]}, l_{[1,j-1]})).\end{aligned}$$

Therefore we get Forge-RSA $\equiv_{tt}^{fp}$ RSA. □

Theorem 8: Exclude-RSA $\equiv_{tt}^{fp}$ RSA

Proof: First we show that Exclude-RSA $\le_{tt}^{fp}$RSA. For inputs $(n_{j,[1,l]}, r_{[1,l]}, e_{j,[1,l]}, H_{[1,j]}, p_{[1,j-1]}, q_{[1,j-1]}, d_{[1,j-1],[1,l]}, sgn_{[1,j]}, l_{j-2}, l_j)$ of Exclude-RSA, set $T = sgn_{j-2} \oplus H_j$, and set the minimum integer l_j such that $n_{j,l_j} > T$. If such a l_j does not exist, then output $\perp$. Then

$$\begin{aligned}&\texttt{Exclude-RSA}(n_{j,[1,l]}, r_{[1,l]}, e_{j,[1,l]}, H_{[1,j]}, p_{[1,j-1]},\\ &\quad q_{[1,j-1]}, d_{[1,j-1],[1,l]}, sgn_{[1,j]}, l_{j-2}, l_j)\\ &= (\texttt{RSA}(n_{j,l_j}, T, e_{j,l_j}), l_j).\end{aligned}$$

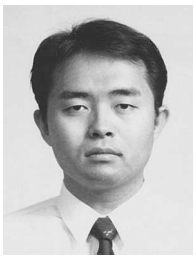
Next we show that RSA $\le_{1-tt}^{fp}$ Exclude-RSA. For inputs (n, m, e) of RSA, set $r_1 = 1$, $n_j = n_{j,1} = n$, $e_{j,1} = e$, fix $e_{j,[2,l]} \in \mathbb{Z}_n$, primes $p_{[1,j-1]}$, $q_{[1,j-1]}$, and $r_{[2,l]}$, computes key pairs of $(n_{[1,j-1],[1,l]}, d_{[1,j-1],[1,l]}, e_{[1,j-1],[1,l]})$ from $p_{[1,j-1]}$, $q_{[1,j-1]}$, and $r_{[1,l]}$. Then fix $H_{[1,j-1]} \in \mathbb{Z}_{n_{min}}$ for $n_{min} = \min\{n_i\}_i$, and set

$$\begin{aligned}&(sgn_{[1,j-1]}, l_{[1,j-1]})\\ &\quad = \texttt{SIGN-RSA}(n_{[1,j-1],[1,l]}, H_{[1,j-1]}, d_{[1,j-1],[1,l]},\\ &\qquad e_{[1,j-1],[1,l]}),\end{aligned}$$

and $H_j = m \oplus sgn_{j-2}$. Then

$$\begin{aligned}&\texttt{RSA}(n, m, e)\\ &\quad = \psi_1(\texttt{Forge}-\texttt{RSA}(n_{j,[1,l]}, r_{[1,l]}, e_{j,[1,l]}, H_{[1,j]},\\ &\qquad p_{[1,j-1]}, q_{[1,j-1]}, d_{[1,j-1],[1,l]}, sgn_{[1,j]}, l_{j-2}, l_j).\end{aligned}$$

Therefore we get $\texttt{Exclude-RSA} \equiv_{tt}^{fp} \texttt{RSA}$. □



Shirow Mitomi received the B.E. from the Department of Computer Science, Tokyo Institute of Technology and the M. Info. Sc. from JAIST in 1998 and 2000 respectively. He had researched the security of flexible multiple signature systems. He has joined FUJITSU LIMITED since 2000 and developes Internet banking systems.

Atsuko Miyaji received the B.Sc., the M.Sc., and Dr.Sci. degrees in mathematics from Osaka University, Osaka, Japan in 1988, 1990, and 1997 respectively. She joined Matsushita Electric Industrial Co., LTD from 1990 to 1998 and engaged in research and development for secure communication. She has been an associate professor at JAIST (Japan Advanced Institute of Science and Technology) since 1998. Her research interests include the application of projective varieties theory into cryptography and information security. She is a member of the International Association for Cryptologic Research and the Information Processing Society of Japan.