A Multisignature Scheme with Message Flexibility, Order Flexibility and Order Verifiability

Shirow Mitomi and Atsuko Miyaji

School of Information Science, Japan Advanced Institute of Science and Technology
miyaji@jaist.ac.jp

Abstract. 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([4, 13, 11, 8, 1]). They are classified into two types: RSA([9])-based multisignature([4, 8]), and discrete logarithm problem(DLP) based multisignature([13, 11, 1]), all of which assume that a message is fixed beforehand. In a sense, these protocols do not have a feature of message flexibility. Furthermore all schemes which satisfy with order verifiability designate order of signers beforehand [13, 1]. 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 multisignature scheme with flexibility and verifiability. We also present two practical schemes based on DLP based message recover signature([7]) and RSA signature([4]), respectively.

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([4, 13, 11, 8, 1]). They are classified into two types: RSA([9]) based multisignature([4, 8]), and discrete logarithm problem(DLP) based multisignature([13, 11, 1]). 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([4, 13, 11, 8, 1]), 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 [4, 11, 8] have a feature of *order verifiability* neither. Order verifiability is first realized in [13, 1]. 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 [1]: 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([7]) and RSA signature([4]). 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, 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 multisignature scheme 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([1]) 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([1]).

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 schemes with flexibility and verifiability for both message and order. 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 denoted $m_1 = Patch(M_1)$. We use a signature scheme with message recovery feature. The signature generation or message recovery function is denoted by $Sign(sk_i, m_i) = sgn_i$, or $Rec(pk_i, sgn_i)$, 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. 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. Figure 1 and 2 show the signature generation and verification, respectively.

Signature generation:

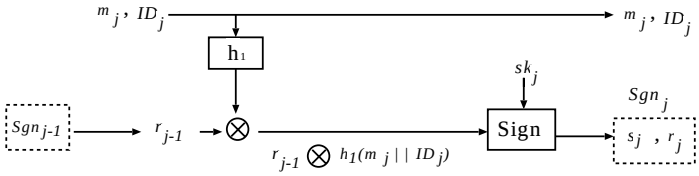

Fig. 1. I_j's signature generation

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,

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

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

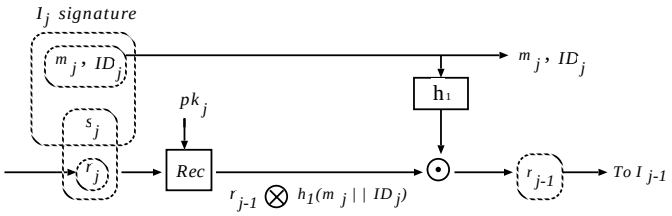

Fig. 2. I_j's signature verification step

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 three 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 next section, we present two schemes based on DLP and RSA.

4 Two concrete multisignature schemes

In this section, we give two examples based on DLP and RSA.

4.1 DLP based scheme

There are many variants of DLP based schemes in both types of message with appendix([3, 12, 2]) and message recovery signature([6, 7]). For the sake of convenience, here we uses the message recovery signature scheme with DSA-signature equation([7]). Apparently any message recovery signature scheme can be applied to our multisignature 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 $R_1 = g^{k_1} \pmod p$, $r_1 = (h_1(m_1||ID_1))^{-1} \cdot R_1 \pmod q$, and $s_1 = (x_1 r_1 + 1)k_1^{-1} \pmod q$, 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 $M_{1,2,\cdots,j-1} = Patch(m_1, m_2, \cdots, m_{j-1})$, 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 $R_j = g^{k_j} \pmod p$, $r_j = (h_1(m_j||ID_j) \times r_{j-1})^{-1} \cdot R_i \pmod q$, and $s_j = (x_j r_j + 1)k_j^{-1} \pmod q$, where I_j's signature on m_j is (r_j, s_j).
3. A multisignature of $M_{1,2,\cdots,i} = Patch(m_1, m_2, .., m_i)$ 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 $R'_j = g^{s_j^{-1}} y_j^{r_j \cdot s_j^{-1}} \pmod p$, $T_j = R'_j \cdot r_j^{-1} \pmod q$, 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 $T_1 = R'_1 \cdot r_1^{-1} \pmod q$, 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.

4.2 RSA based scheme

Here we present our multisignature scheme based on RSA multisignature([4]).
Initialization: An authenticated center publishes small primes $\{r_l\} = \{2, 3, 5, \cdots\}$. 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

$n_{i,l} = p_i q_i r_l$, $L_{i,l} = LCM((p_i - 1), (q_i - 1), (r_l - 1))$, $e_{i,l} d_{i,l} = 1 \pmod{L_{i,l}}$, by using $\{r_l\}$. Signer I_i publishes all his public keys $n_{i,l}$, $e_{i,l}$ and r_l like Table 1.

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:

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 1. I_i's pairs of secret key and public key

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 of $M_{1,2,...,i} = Patch(m_1, m_2, .., m_i)$ 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 $T' = (sgn_j)^{e_{j,l_j}} \pmod{n_{j,l_j}}$, and $sgn_{j-1} = h_1(m_j||ID_j) \oplus T'$ by using I_j's public key (n_{j,l_j}, e_{j,l_j}). Let $j = j - 1$ and repeat step2.
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 $T' \stackrel{?}{=} h_1(m_1||ID_1)$.

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.

4.3 Performance evaluation

We evaluate our two multisignature schemes 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. For a simple discussion, we assume the following conditions: 1. a primitive arithmetic of binary methods([5]) 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 and q are set to 1024 and 160 bits respectively, in DLP-based signature schemes; 4. 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.

DLP based-multisignature schemes are mainly classified into two types, one-round scheme([11]) and two-round scheme in Section 2. Generally, the signature

	Computation amount $\#M(1024)$		Signature size (bits)	$\#rounds$	Features
	I_i's signature generation	signature verification			
Our scheme	243	$483i$	$160(i+1)$	1	MF, OF, OV
Primitive scheme	242	$483i$	$320i$	1	MF
Scheme([11])	242	$481+241i$	$160+1024i$	1	—
Scheme([1])	483	1778	2048	2	OV

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

Table 2. Performance of DLP-based multisignature schemes

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

Table 3. Performance of RSA based signatures

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([11]) and two-round scheme([1]) Table 2 shows performance of 4 schemes. From Table 2, we see that only the computation amount for signature verification increases, and the signature size is even reduced, compared with the same one-round multisignature. Therefore our protocol can realize three features with message flexibility, order flexibility, and order verifiability only with negligible additional computation amount in signature generation.

Here we compare our RSA-based multisignature scheme with the primitive scheme. Table 3 shows performance of two schemes. From Table 3, we see that our protocol can realize three features, message flexibility, order flexibility, and order verifiability, with neither additional computation amount nor signature size.

5 Security consideration

In this section, we discuss the security relation between our DLP based multisignature scheme and DLP. 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 $m_1, \cdots, m_n \in \mathbb{Z}$ 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 $x_{[1,n]} \in \mathbb{Z}_q$. In our security proof, we use the polynomial-time truth-table($\leq^{fp}_{k-tt}$) reducibility of the function version([10]), which discusses passive attacks. In $\leq^{fp}_{k-tt}$ only k non-adaptive queries to an oracle are allowed.

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. $\mathtt{Forge}(y_n, g, p, q, m_{[1,n]}, ID_{[1,n]}, x_{[1,n-1]}, s_{[1,n-1]}, r_{n-1}, k_n)$ *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}, $x_{[1,n-1]}$, $k_n \in \mathbb{Z}_q^*$, $m_{[1,n]}$, $ID_n \in \mathbb{Z}$, *outputs* $(r_n, s_n) \in \mathbb{Z}_q^* \times \mathbb{Z}_q^*$ *such that* $t_j = g^{s_j^{-1}} y_j^{r_j \cdot s_j^{-1}} \pmod p$, $T_j = t_j \cdot r_j^{-1} \pmod q$, *and* $r_{j-1} = T_j \cdot (h_1(m_j||ID_j))^{-1} \pmod q$ *for* $j = n, n-1, ..., 3, 2$ *and that* $t_1 = g^{s_1^{-1}} y_1^{r_1 \cdot s_1^{-1}} \pmod p$ *and* $T_1 = t_1 \cdot r_1^{-1} \pmod q$ *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. $\mathtt{Exclude}(y_n, g, p, q, m_{[1,n]}, ID_{[1,n]}, x_{[1,n-1]}, s_{[1,n]}, r_n)$ *is the function that on input two primes* p, q *with* $q|p-1$, g, $y_n \in \mathbb{Z}_p^*$, $m_{[1,n]}$, $ID_{[1,n]} \in \mathbb{Z}$, $x_{[1,n-1]}$, r_n, $s_{[1,n]} \in \mathbb{Z}_q^*$, *output* $(s'_n, k_n) \in \mathbb{Z}_q^* \times \mathbb{Z}_q^*$ *such that* $R_n = g^{k_n} \pmod p$, $r'_n = (h_1(m_n||ID_n) \times r_{n-2})^{-1} R_n \pmod q$, *and* $s'_n = (x_n r'_n + 1) k_n^{-1} \pmod q$, *for* $j = n-2, \cdots, 2$: $t_j = g^{s_j^{-1}} y_j^{r_j \cdot s_j^{-1}} \pmod p$, $T_j = t_j \cdot r_j^{-1} \pmod q$, *and* $r_{j-1} = T_j \cdot (h_1(m_j||ID_j))^{-1} \pmod q$, *and that* $t_1 = g^{s_1^{-1}} y_1^{r_1 \cdot s_1^{-1}} \pmod p$, $T_1 = t_1 \cdot r_1^{-1} \pmod q$ *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 $k_{[1,n-1]}$ for 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. $\mathtt{SWAP} \equiv^{fp}_{1-tt} \mathtt{Exclude}$.

For the sake of the following proof, we define the function SIGN that generates a valid signature $(r_n, s_{[1,n]})$ 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 4. $\mathtt{SIGN}(g, p, q, x_{[1,n]}, k_{[1,n]}, m_{[1,n]}, ID_{[1,n]})$ *is the function that on input two primes* p, q *with* $q|(p-1)$, $g \in \mathbb{Z}_p^*$, $x_{[1,n]}, k_{[1,n]} \in \mathbb{Z}_q^*$, $m_{[1,n]}, ID_{[1,n]} \in \mathbb{Z}$, *output* $r_n, s_{[1,n]} \in \mathbb{Z}_q^*$ *such that for* $j = n, ..., 3, 2$, $t_j = g^{s_j^{-1}} y_j^{r_j \cdot s_j^{-1}} \pmod p$, $T_j = t_j \cdot r_j^{-1} \pmod q$ *and* $r_{j-1} = T_j \cdot (h_1(m_j || ID_j))^{-1} \pmod q$ *and that* $t_1 = g^{s_1^{-1}} y_1^{r_1 \cdot s_1^{-1}} \pmod p$, $T_1 = t_1 \cdot r_1^{-1} \pmod q$ *if such* $r_n, s_{[1,n]} \in \mathbb{Z}_q$ *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 2. $\mathtt{Forge} \equiv^{fp}_{1-tt} \mathtt{DLP}$

proof: First we show that $\mathtt{Forge} \leq^{fp}_{1-tt} \mathtt{DLP}$. For inputs $(y_n, g, p, q, m_{[1,n]}, ID_{[1,n]}, x_{[1,n-1]}, s_{[1,n-1]}, r_{n-1})$ of Forge, fix $k_n \in \mathbb{Z}_q$ and set $R_n = g^{k_n} \pmod p$, $r_n = r_{n-1}^{-1} \cdot h_1(m_n || ID_n)^{-1} \cdot R_n \pmod p$. Then
$\mathtt{Forge}(y_n, g, p, q, m_{[1,n]}, ID_{[1,n]}, x_{[1,n-1]}, s_{[1,n-1]}, r_{n-1}, k_n)$
$= (r_n, (\mathtt{DLP}(y_n, g, p, q) r_n + 1) k_n^{-1} \pmod q)$.
$= (r_n, s_n)$. ∎
Next we show that $\mathtt{DLP} \leq^{fp}_{1-tt} \mathtt{Forge}$. For input (y_n, g, p, q) of DLP, fix $k_{[1,n]} \in \mathbb{Z}_q^*$, $m_{[1,n]}, ID_{[1,n]} \in \mathbb{Z}$, $x_{[1,n-1]} \in \mathbb{Z}_q^*$, and set
$(r_{n-1}, s_{[1,n-1]}) = \mathtt{SIGN}(g, p, q, x_{[1,n-1]}, k_{[1,n-1]}, m_{[1,n-1]}, ID_{[1,n-1]})$,
which is computed in time polynomial from the definition. Then
$\mathtt{DLP}(y_n, g, p, q)$
$= (\psi_2(\mathtt{Forge}(y_n, g, p, q, m_{[1,n]}, ID_{[1,n]}, x_{[1,n-1]}, s_{[1,n-1]}, r_{n-1}, k_n)) \cdot k_n - 1) r_n^{-1}$,
where $r_n = \psi_1(\mathtt{Forge}(y_n, g, p, q, m_{[1,n]}, ID_{[1,n]}, x_{[1,n-1]}, s_{[1,n-1]}, r_{n-1}, k_n))$ and $y_n = g^{x_n}$. ∎
Therefore we get $\mathtt{DLP} \equiv^{fp}_{1-tt} \mathtt{Forge}$. ∎

Theorem 3. $\mathtt{Exclude} \equiv^{fp}_{1-tt} \mathtt{DLP}$

proof: First we show that $\mathtt{Exclude} \leq^{fp}_{1-tt} \mathtt{DLP}$. For inputs $(y_n, g, p, q, m_{[1,n]}, ID_{[1,n]}, x_{[1,n-1]}, s_{[1,n]}, r_n)$ of Exclude, fix $k_n \in \mathbb{Z}_q$, and set $R_n = g^{k_n} \pmod p$, and $r'_n = r_{n-2}^{-1} \cdot h_1(m_n || ID_n)^{-1} \cdot R_n \pmod p$. Then
$\mathtt{Exclude}(y_n, g, p, q, m_{[1,n]}, ID_{[1,n]}, x_{[1,n-1]}, s_{[1,n]}, r_n, k_n)$
$= ((\mathtt{DLP}(y_n, g, p, q) r'_n + 1) k_n^{-1} \pmod q, k_n)$ ∎
Next we show that $\mathtt{DLP} \leq^{fp}_{1-tt} \mathtt{Exclude}$. For inputs (y_n, g, p, q) of DLP, fix $k_{[1,n-1]} \in \mathbb{Z}_q^*$, $m_{[1,n]}, ID_{[1,n]} \in \mathbb{Z}$, $x_{[1,n-1]} \in \mathbb{Z}_q^*$, and set

$(r_{n-2}, s_{[1,n-2]}) = \mathtt{SIGN}(g, p, q, x_{[1,n-2]}, k_{[1,n-2]}, m_{[1,n-2]}, ID_{[1,n-2]})$,
which is computed in time polynomial from the definition. Then
$\mathtt{DLP}(y_n, g, p, q) = (s'_n \cdot k_n - 1) \cdot {r'}_n^{-1}$, where
$s'_n = \psi_1(\mathtt{Exclude}(y_n, g, p, q, m_{[1,n]}, ID_{[1,n]}, x_{[1,n-1]}, s_{[1,n]}, r_n))$,
$k_n = \psi_2(\mathtt{Exclude}(y_n, g, p, q, m_{[1,n]}, ID_{[1,n]}, x_{[1,n-1]}, s_{[1,n]}, r_n))$,
$R_n = g^{k_n} \pmod p$, and $r'_n = (r_{n-2} \cdot h_1(m_n || ID_n))^{-1} \cdot R_n \pmod q$. ∎
Then we get $\mathtt{DLP} \equiv^{fp}_{1-tt} \mathtt{Exclude}$.∎

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 nd 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.
Signature generation:

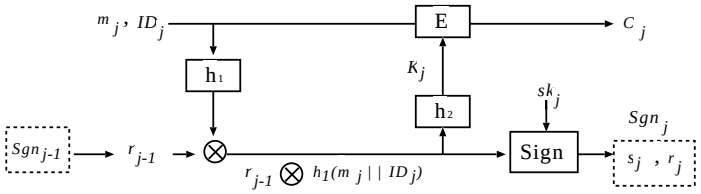

Fig. 3. I_j's signature generation

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 Section 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,

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

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

2. A signer I_j verifies the signature from I_{j-1}, $m_1, \cdots, m_{j-1}$ according to the verification step in the next page, 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

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

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

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).

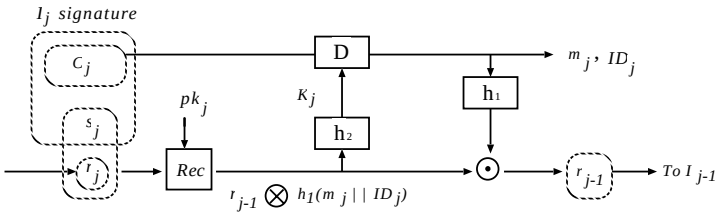

Fig. 4. I_j's signature verification step

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 three features, Message flexibility, Order flexibility and Order verifiability, maintaining both signature size and computation amount in signature generation/verification low: only the computation amount for the signature verification increases, 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 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. Burmester, Yvo Desmedt, Hiroshi Doi, Masahiro Mambo, Eiji Okamoto, Mitsuru Tada, and Y. Yoshifuji, "A Structured ElGamal-Type Multisignature Scheme", *Advances in Cryptology-Proceedings of PKC'2000*, Lecture Notes in Computer Science, (2000), Springer-Verlag, 466-482.
2. "Specification for a digital signature standard", National Institute for Standards and Technology, Federal Information Standard Publication XX, draft (1991).

3. T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms", *IEEE Trans. Inform. Theory*, Vol. IT-31 (1985), 469-472.
4. K. Itakura and K. Nakamura, "A public-key cryptosystem suitable for digital multisignatures". *NEC J.Res.Dev.71*(Oct.1983).
5. D. E. Knuth, *The art of computer programming, vol. 2, Seminumerical Algorithms*, 2nd ed., Addison-Wesley, Reading, Mass. 1981.
6. A. Miyaji, "Another countermeasure to forgeries over message recovery signature", *IEICE Trans.*, Fundamentals. vol. E80-A, No.11(1997), 2192-2200.
7. K. Nyberg and R. A. Rueppel, "Message recovery for signature schemes based on the discrete logarithm problem", *Designs Codes and Cryptography*, **7**(1996), 61-81.
8. T. Okamoto, "A digital Multisignature Scheme Using Bijective Public-key Cryptosystems", *ACM Trans. on Computer Systems*, Vol.6, No.8(1988), 432-441.
9. R. Rivest, A. Shamir and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", *Communications of the ACM*, vol.21, No.2(1978), 120-126.
10. K. Sakurai and H. Shizuya "Relationships among the computational powers of breaking Discrete Log cryptosystem", *Advavnced in Cryptology-Proceedings of Eurocrypt'95*, Lecture Notes in Computer Science, **921**(1995), Springer-Verlag, 341-355. (J. Cryptology,**11** (1998), 29-43.)
11. A. Shimbo, "Multisignature Schemes Based on the Elgamal Scheme", *The 1994 Symposium on Cryptography and Information Security*, **SCIS94-2C**, Jan. 1994.
12. C. P. Schnorr, "Efficient signature generation by smart cards", *Journal of cryptology*, **4**(1991), 161-174.
13. T. Saito, "A multiplesignature Scheme Enabling a Specified Signer's Order", *The 1997 Symposium on Cryptography and Information Security*, **SCIS97-33A**, Jan. 1994.