Structured multisignature algorithms

L. Harn, C.-Y. Lin and T.-C. Wu

Abstract: A structured multisignature scheme is an order-sensitive multisignature scheme that allows participating signers to sign messages in compliance with a specified signing order. It has been shown that the Burmester et al. order-sensitive multisignature scheme cannot prevent all signers producing a valid multisignature without following the specified signing order. The paper proposes two structured multisignature algorithms, one based on the RSA scheme and the other on an ElGamal-type scheme. Incorporation of both order-free and order-sensitive multisignature algorithms together is shown to construct a generalised multisignature algorithm.

1 Introduction

The multisignature is a kind of group-oriented cryptography that was first introduced by Desmedt [1] in 1987. The group has a security policy that requires a multisignature to be signed by all group members with the knowledge of multiple private keys. However, to verify the multisignature can be verified using a corresponding group public key. In general, we assume that all group members do not trust each other. On the other hand, if they do trust each other, all private keys can be shared among themselves. Thus the multisignature is identical to a normal digital signature. An efficient digital multisignature scheme can combine all individual signatures of the same message into a single multisignature and this multisignature can be verified efficiently.

Harn [2] has proposed an ElGamal-type multisignature scheme that can combine all individual signatures into a multisignature without any data expansion. In other words, the length of the multisignature is equivalent to the length of each individual signature. This result is reasonable since the length of the signature/multisignature depends only on the security assumptions of signature schemes and not on the number of signers involved. Multiple signers with knowledge of multiple private keys can produce a fixed length of multisignature. Harn’s multisignature scheme is order-free since signers can sign the message in any order.

Burmester et al. [3] proposed a structured ElGamal-type multisignature scheme in the PKC 2000 conference. A structured multisignature scheme is an order-sensitive multisignature scheme that only allows participating signers to sign messages in compliance with a specified signing order. However, in a recent paper [4] it has been shown that, the Burmester et al. order-sensitive multisignature scheme cannot prevent all participating signers producing a valid multisignature without following the specified signing order. We do not consider that this is a serious setback since it requires all signers to co-operate. This condition contradicts the general assumption of group-oriented cryptography.

We propose two structured multisignature algorithms. One algorithm is based on the RSA scheme [5] and the other on an ElGamal-type signature scheme [6]. Our solution has better performance than the solution proposed by Burmester et al. In the Burmester et al. solution, all signers need to follow the signing order twice to obtain a group signature; however, in our solution, one-round processing is required. In addition, there are fewer computations needed by each signer. We also show the incorporation of both order-free and order-sensitive multisignature algorithms together to construct a generalised multisignature algorithm.

2 Structured multisignature algorithm based on RSA scheme

We assume throughout this paper that there are t signers $U_1, U_2, \ldots, U_t$ in a group. The specified signing order is $\langle U_1, U_2, \ldots, U_t \rangle$. Each signer needs to follow the RSA scheme to select two large secret primes $p_i$ and $q_i$ and publish their product $n_i$. At the same time, each signer needs to determine the public key $e_i$ and private key $d_i$ accordingly. However, to construct an order-sensitive multisignature scheme their publicly-known products $n_1, n_2, \ldots, n_t$ need to satisfy the following requirement, that $n_1 < n_2 < \ldots < n_t$. To sign a message $m$, $U_1$ computes $S_1 = h(m)^{e_1} \mod n_1$ and sends it to $U_2$, where $h()$ is a one-way hash function; $U_2$ computes $S_2 = S_1^{e_2} \mod n_2$ and sends it to $U_3$, and so on. The multisignature is the output of the last signer $S_t$. To verify this multisignature the verifier needs to reverse the signing order to check whether $h(m)$ is identical to $(\ldots ((S^t_i \mod n_1)^{e_{t-1}} \mod n_{t-1}) \ldots)^{e_1} \mod n_1$.

This algorithm is order-sensitive because the commutative law does not apply to modular multiplication that involves two different moduli $n_i$ and $n_j$ with $gcd(n_i, n_j) = 1$. That is, $a \mod n_j \mod n_i$ is not $(a \mod n_j) \mod n_i$, where $a > n_j$, and $a > n_i$. Thus if a multisignature is signed by signers without following the specified order, the multisignature cannot be verified successfully.
3 Structured multisignature algorithm based on ElGamal-type scheme

3.1 Public parameters

A large prime \( p \), where \( p = 2q + 1 \) and \( q \) is also a prime, and a primitive element \( \alpha \) of \( \text{GF}(p) \) are known to all signers.

3.2 Generating individual and group private/public key pairs

Initially all signers need to work together to generate their public keys \( y_i \) for \( i = 1, 2, \ldots, t \), and their group public key \( y \). Each user randomly selects an odd private key \( x_i \) from \([1, q - 1]\). The last signer \( U_t \) computes \( y_t = \alpha^{x_t} \mod p \) and sends it to \( U_{t-1} \); \( U_{t-1} \) computes \( y_{t-1} = y_t^{x_{t-1}} \mod p \) and sends it to \( U_{t-2} \), and so on. In other words \( y_i = y_{i+1}^{x_{i+1}} \mod p \) for \( i = 1, 2, \ldots, t \) where \( y_{t+1} = \alpha \). \( y_i \) is the public key of the signer \( U_i \). The group public key \( y \) is the public key of the first signer \( U_1 \) such that \( y = y_1 \), where \( y_1 = \alpha^{x_1} \mod p \). The group private key is \( x_1 x_2 \cdots x_t y_1 \mod p \), which involves all signers’ private keys. Figure 1 shows the signing sequence and key generation order and related information of each signer. It is important to know that each signer \( U_i \) needs to prove to all others knowledge of the private key \( x_i \) before all other signers accepting the revealed value \( y_i \) as \( U_i \)’s public key. In case a digital certificate is associated with each public key, each signer needs to prove the knowledge of secret key to the certificate authority (CA) before obtaining a digital certificate from the CA. This procedure can prevent some possible attack as pointed out by Langford [7].

3.3 Generating individual signatures

To sign an ElGamal-type signature there is a pair of short-term private and public keys computed by each signer. This computation is independent of the message and can be precomputed. Similar to the order-free multisignature algorithm proposed in [2], each signer \( U_i \) randomly selects a short-term private key \( k_i \) from \([1, q - 1]\) and computes \( r_i = y_i^{k_i} \mod p \), where \( y_i = \alpha^{x_i} \). After receiving all \( r_i \) for \( i = 1, 2, \ldots, t \), each signer can compute \( R = r_1 r_2 \cdots r_t \mod p \). We mention again that since this process is independent of the message, it does not need to follow the specified signing order and it can be precomputed.

For a given message \( m \) where \( m \) is the one-way hash of the message, following the specified signing order \( \langle U_1, U_2, \ldots, U_t \rangle \) each signer computes an individual signature \( s_i \) that satisfies the equation \( x_i s_i - 1 = k_i R + s_i \mod p \), where \( s_0 = m; s_t \) is sent to the next signer.

3.4 Verifying individual signature

On receiving the individual signature \( s_i \) from the preceding signer \( U_i \) the current signer \( U_{i+1} \) needs to verify that all preceding signers \( \langle U_1, U_2, \ldots, U_i \rangle \) have signed the message \( m \) properly. Since all preceding signers’ individual signatures satisfy the following equations:

\[
\begin{align*}
y_1^m &= r_1 y_2^m \mod p \\
y_2^m &= r_2 y_3^m \mod p \\
&\vdots \\
y_{i-1}^m &= r_i y_i^m \mod p \\
y_i^m &= r_1 y_2^m \cdots y_i^m \mod p 
\end{align*}
\]

by multiplying all these equations together we obtain the following verification equation as:

\[
y_1^m = (r_1 r_2 \cdots r_i) y_i^m \mod p
\]

We claim that the signer \( U_{i+1} \) can use this verification equation to verify that all preceding signers \( \langle U_1, U_2, \ldots, U_i \rangle \) have signed the message \( m \) properly.

3.5 Generating group signature

We claim that \( (R, s) \) is the multisignature of the message \( m \).

3.6 Verifying multisignature

Similarly, by multiplying all \( i \) equations together we obtain

\[
\begin{align*}
y_1^m &= R y_1^m \mod p \\
y_2^m &= R y_2^m \mod p \\
&\vdots \\
y_i^m &= R y_i^m \mod p 
\end{align*}
\]

We claim that any verifier can access the group public key \( y \) to verify the multisignature \( (R, s) \) of message \( m \), according to (2).

---

**Fig. 1** Signing order of a structured multisignature scheme and related information of each signer

<table>
<thead>
<tr>
<th>Private key</th>
<th>Public key</th>
<th>Public parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_i )</td>
<td>( y_i = \alpha^{x_i} \mod p )</td>
<td>( y_1 = \alpha^{x_1} \mod p )</td>
</tr>
<tr>
<td>( x_2 )</td>
<td>( y_2 = \alpha^{x_2} \mod p )</td>
<td>( y_2 = \alpha^{x_2} \mod p )</td>
</tr>
<tr>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
</tr>
<tr>
<td>( x_{t-1} )</td>
<td>( y_{t-1} = \alpha^{x_{t-1}} \mod p )</td>
<td>( y_{t-1} = \alpha^{x_{t-1}} \mod p )</td>
</tr>
<tr>
<td>( x_t )</td>
<td>( y_t = \alpha^{x_t} )</td>
<td>( \alpha )</td>
</tr>
</tbody>
</table>
3.7 Security analysis

The security of this proposed scheme is based on the computational assumption of discrete logarithm (DL) problem. Here we list some security features of this algorithm.

- The group public key $y$, where $y_1 = y^{x_1} \mod p$, corresponds to a private key $x_1, x_2, \ldots, x_n, y \mod p - 1$, which involves all signers’ private keys. Finding the private key from the public key is equivalent to solving the DL problem.

- Although the group public key $y$ is identical to the first signer’s public key $y_1$, the first signer only generates a signature pair $(R, s)$ that satisfies
  \[ y_1^m = R^s y_2^m \mod p \]
  and this verification equation is different from (2).

- All signers must follow the specified order exactly to obtain a valid multisignature. The absence of a single individual signature results in an invalid multisignature.

- Since $y_1 = y^{x_1} \mod p$, which involves private keys $x_1, x_2, \ldots, x_n$, we rewrite (1) as
  \[ y_1(x_1, \ldots, x_n) = (r_1 r_2 \ldots r_i)^{R} y_1^{x_i} \mod p \]

Forging an individual signature $x_i$ to satisfy (1) is equivalent to solving the DL problem. The individual signature verification (1) enables the signer $U_i$ to verify that all preceding signers ($U_1, U_2, \ldots, U_i$) have sign the message $m$ properly.

- Since our algorithm enables one to verify any individual signature in the middle of the signing sequence, the signing process can be halted once any invalid individual signature has been found.

- To achieve maximal security each signer’s public key $y_i$ should be a primitive element of $GF(p)$. According to the following lemma, we show that this condition is guaranteed since each private key is an odd integer from $\{1, q - 1\}$.

3.8 Lemma 1

Let $x$ be a primitive element of $GF(p)$. The set $\Gamma$

\[ \Gamma = \{x^{2i+1} \mod q \mid i > 0, \ 2i + 1 < q\} \]

consists of all primitive elements of $GF(p)$ and all quadratic nonresidue modulo $p$ except for $-1 = x^q$.

3.9 Discussion

In a previous Section we have shown that for a specific signing order ($U_1, U_2, \ldots, U_t$) that involves $t$ signers, the public key is $y = y^{x_1} \mod p$, where $x_i$ is the private key of signer $U_i$. The multisignature is verified based on the group public key $y$ and public parameters $x$ and $p$. Actually, for any subset of signing sequence from the original signing sequence ($U_1, U_2, \ldots, U_t$), the structured multisignature can be generated in a similar way. For example, assume that an order-sensitive signing sequence ($U_{i-1}, U_i, U_{i+1}$) that involved three signers, each multisignature can be verified based on the group public key $y_{i-1}$ and public parameters $y_{i+2}$ and $p$, where $y_{i-1} = y^{x_{i-1}x_{i+1}} \mod p$.

4 Generalised multisignature scheme

A generalised multisignature algorithm should work for applications that contain both order-free and order-sensitive cases. For example, as shown in Fig. 2, in an order-sensitive sequence ($U_1, U_2, \ldots, U_t$), the signer $U_i$ always acts as $n$ individual signers $U_1, U_2, \ldots, U_t$, at $U_i$ level, these $n$ signers $U_1, U_2, \ldots, U_t$ generate order-free multisignature. At $U_i$ level, these $n$ signers $U_1, U_2, \ldots, U_t$ generate order-sensitive multisignature. Here we incorporate the order-free multisignature in [2] and the

Fig. 2 Signing order and related information of each signer for generalised multisignature
order-sensitive multisignature algorithm proposed in the previous Section to construct an efficient solution.

Following the same procedure as described, signers follow the specific order \( \{U_t, U_{t-1}, \ldots, U_{t+1}\} \) to generate their private keys and public keys as shown in Fig. 2. At signer \( U_{t,j} \), according to [2], each signer \( U_{t,j} \) randomly selects an odd private key \( x_{t,j} \) from \([1, q - 1]\) and computes the public key as \( y_{t,j} = y_{t-1}^{x_{t,j}} \mod p \). The private key \( x_i \) of signer \( U_i \) is determined by all signers, \( U_{t,j} \) for \( j = 1, 2, \ldots, n \) as \( x_i = x_{t,1} + x_{t,2} + \cdots + x_{t,j} \mod p - 1 \) and the public key is \( y_i = y_{t,1}^{x_i} \mod p \). Then, the remaining signers follow the specific order \( \{U_{t-1}, U_{t-2}, \ldots, U_1\} \) to generate their private and public keys.

To sign an ElGamal-type signature each signer \( U_{t,j} \) at the signer \( U_t \) in the order-free sequence randomly selects a short-term private key \( k_{t,j} \) from \([1, q - 1]\) and computes \( r_{t,j} = r_{t-1}^{y_{t,j}} \mod p \). After receiving all \( r_{t,j} \) for \( j = 1, 2, \ldots, n \) each signer can compute \( r_i = r_{t-1}^{x_i} \mod p \). Similarly, each signer \( U_{t,j} \) at the system level, in the order-sensitive sequence randomly selects a short-term private key \( k_j \) from \([1, q - 1]\) and computes \( r_j = y_{t,j}^{k_j} \mod p \), where \( y_{t,j} = x_j \). After receiving all \( r_j \) for \( i = 1, 2, \ldots, t \) each signer, at the system level, can compute \( R = r_1 r_2 \cdots r_t \mod p \).

For a given message \( m \), where \( m \) is the one-way hash of the message, signers in the order-sensitive sequence \( \{U_1, U_2, \ldots, U_{t-1}\} \) compute an individual signature \( s_j \) as described previously. Then each signer in the order-free sequence at \( U_t \) computes an individual signature \( s_{t,j} \) that satisfies the equation \( x_{t,j} s_{t-1} = k_{t,j} R + s_{t,j} \mod p - 1 \). These individual signatures satisfy the following equations:

\[
\begin{align*}
    y_{t-1}^{s_{t,1}} &= r_{t-1}^{x_{t,1}} y_{t-1}^{s_{t,1}} \mod p \\
    y_{t,2}^{s_{t,2}} &= r_{t-1}^{x_{t,2}} y_{t-1}^{s_{t,2}} \mod p \\
    &\vdots \\
    y_{t,n}^{s_{t,n}} &= r_{t-1}^{x_{t,n}} y_{t-1}^{s_{t,n}} \mod p 
\end{align*}
\]

Multiplying these equations together obtains

\[
y_{t-1}^{s_1 + s_2 + \cdots + s_n} = r_{t-1}^{x_1 + x_2 + \cdots + x_n} \mod p \]

Then the rest of the signers in the order-sensitive sequence follow the specific order \( \{U_{t+1}, U_{t+2}, \ldots, U_t\} \) to generate their individual signatures.

5 Conclusions

We have proposed two order-sensitive multisignature schemes, one based on the RSA scheme and the other based on an ElGamal-type scheme. Both schemes are very efficient in terms of the length of multisignature and verification time. In addition, both algorithms work without the assistance of a mutually trusted third party. We also show the incorporation of both order-free and order-sensitive multisignature algorithms together to construct a generalised multisignature algorithm.

6 References