Meta-Multisignature schemes based on the discrete logarithm problem

Patrick Horster, Markus Michels and Holger Petersen

Theoretical Computer Science and Information Security, University of Technology Chemnitz-Zwickau, Straße der Nationen 62, D-09111 Chemnitz, Germany
E-mail: {pho,mmi,hpe}@informatik.tu-chemnitz.de

In this paper we present the first multisignature scheme giving message recovery based on the discrete logarithm problem. An efficient multisignature scheme with appendix has been proposed by Harn recently. We cryptanalyze this scheme, present two attacks and show how to countermeasure them. Furthermore we adopt the Meta-ElGamal and Meta-Message recovery signature scheme with one and two message blocks to the slightly modified scheme and give conditions which variants can be used. We show that for the variants giving message recovery it is useful to apply the variants with two message blocks to prevent the described attacks and to guarantee the efficiency of the scheme.

1. Introduction

A multisignature scheme allows multiple signers to sign the same message. A simple solution is that every signer signs the message using a normal signature scheme, but this has the drawback that the data expansion increases with the number of signers. The goal is to design a multisignature scheme without data expansion depending on the number of signers.

Recently a multisignature scheme based on the discrete logarithm problem with this property was proposed by Harn [Har194, Har294]. We give a brief review of this scheme in section 2 and present two attacks and countermeasure them in section 3. Then the Meta-ElGamal (MEG) signature scheme [HMP194, HMP294] and Meta-Message recovery (MMR) signature scheme [HMP394] are briefly described in sections 4 and 5. After this we show in sections 6 and 7 how to generalize the modified multisignature scheme to some variants of the the MEG- and MMR- signature scheme and give conditions which variants can be used. The scheme in section 7 is the first multisignature scheme giving message recovery based on the discrete logarithm problem.

2. The basic multisignature scheme

First we review the underlying signature scheme and how this can be extended to provide multisignatures [Har294].

A trusted third party chooses a large prime p, a generator $\alpha \in \mathbf{Z}_p^*$ and publishes them as system parameters. The signer *Alice* chooses a random number $x_A \in \mathbf{Z}_{p-1}^*$ and computes $y_A := \alpha^{x_A} \pmod{p}$. She publishes y_A and keeps x_A secret. These values are constant for all messages to be signed. To sign a message $m' \in \mathbf{Z}_{p-1}$ Alice computes $m := h(m')$ with a suitable hash function h, chooses a random number $k \in \mathbf{Z}_{p-1}^*$ and computes $r := \alpha^k \pmod{p}$. Then she solves the equation

$$s := x_A(m+r) - k \pmod{p-1}. \tag{1}$$

The triple $(m; r, s)$ is the signed message. It can be verified by checking the congruence

$$r\alpha^s \equiv y_A^{m+r} \pmod{p}. \tag{2}$$

In a multisignature scheme we have t different signers $u_1, \ldots, u_t$. Each of them has a secret key $x_i \in \mathbf{Z}_{p-1}^*$ and a related public key $y_i := \alpha^{x_i} \pmod{p}$. Each signer $u_i, i \in [1:t]$, chooses a random number $k_i \in \mathbf{Z}_{p-1}^*$ and computes $r_i := \alpha^{k_i} \pmod{p}$. Then he broadcasts r_i to all other signers, such that every signer can compute

$$r := \prod_{i=1}^{t} r_i \pmod{p}.$$

Now each signer u_i computes his signature parameter s_i by

$$s_i := x_i(m+r) - k_i \pmod{p-1}.$$

He transmits s_i to a clerk who additionally knows m and r and whose task is to check each individual signature by verifying the congruence

$$y_i^{m+r} \equiv r_i\alpha^{s_i} \pmod{p}.$$

He also generates the multisignature of the message m by computing

$$s := \sum_{i=1}^{t} s_i \pmod{p-1}.$$

The triple (m, r, s) is the multisignature of the message m which can be verified by checking the congruence

$$y^{m+r} \equiv r\alpha^s \pmod{p}, \tag{3}$$

where $y := \prod_{i=1}^{t} y_i \pmod{p}$.

3. Cryptanalysis of the scheme and countermeasures

In this section we present two attacks which threaten the security of the scheme unless modified. The first can be countermeasured easily while the second is somewhat harder but can also be avoided.

3.1 The first attack

The attacker's (Carol) aim is to claim that a signature for a message m which is given by the users Alice, Bob and Donald is additionally signed by Carol and Alice. Thus no verifier can proof which of the persons has really signed the message and nobody trusts the scheme anymore.

The trick is that Carol is a normal user with a public key y_C. She chooses this key not by choosing x_C randomly and computing $y_C := \alpha^{x_C} \pmod p$ but she computes $y_C := y_B y_D \pmod p$. Note that this attack is weak because Carol doesn't know her own secret key and as a result she can't sign any message. But as the congruence

$$y_A y_B y_D \equiv y_A y_C \pmod p$$

and the verification equation (3) of the multisignature scheme hold no verifier can decide if any message was signed by Alice, Bob and Donald or just by Alice and Carol.

With a slight modification this attack can easily be countermeasured: The trusted third party accepts just those public keys y_i where the signer can proof that he knows the discrete logarithm of the public-key using the zero-knowledge-proof due to Chaum, Evertse and van de Graaf [ChEG87]. Obviously this attack does not work anymore now. Another solution, due to one of the anonymous referees, is that every user i signs a fixed message (e.g. "My public key is y_i") and sends the signature to the trusted third party. It is accepted if the verification equation holds and - to avoid a replay attack - y_i was not used before by any other user.

3.2 The second attack

This attack works if two or more signers of a document and the clerk collude. Alice, Bob, Carol, Donald and Eve should sign a document m. Carol, Donald and Eve (the tiger team) collude and choose the parameters $k_C, k_D, k_E \in \mathbf{Z}_{p-1}$ such that

$$k_C + k_D + k_E \equiv 0 \pmod{p-1}.$$

Therefore $r_C r_D r_E \equiv 1 \pmod p$ and $r := r_A r_B r_C r_D r_E \equiv r_A r_B \pmod p$. Furthermore the clerk colludes with them and reveals s_A and s_B. Then the tiger team knows that r and $s := s_A + s_B \pmod{p-1}$ is a signature from Alice and Bob on the message m. Thus no signer has a guarantee that the other signers sign the document in spite of the r_i–broadcasting in the first step of the signature generation.

How can the parameters k_i of the tiger team be generated ? A simple solution is that the members of the tiger team have unconditional trust to each other and they generate them together. Note that the secret key of each member can be computed by the other members if the clerk reveals the signature parameters s_i. With more computational effort the attack can be done by the tiger team without assuming unconditional trust between the team members: Every member $u_i, i \in [1:w]$, of the tiger team chooses the parameter k_i and $l \in \mathbf{Z}_{p-1}^*$ values $b_{ij} \in \mathbf{Z}_{p-1}$ which satisfy

$$\sum_{j=1}^{l} b_{ij} = 0 \pmod{p-1}.$$

Then the values $c_{ij} := k_i + b_{ij} \pmod{p-1}$ are sent through a secure MIX-net [Chau81]. The outputs of the MIX-net are added. We get

$$o := \sum_{i=1}^{w}\sum_{j=1}^{l} c_{ij} = l\sum_{i=1}^{w} k_i \pmod{p-1}$$

Now one member, e.g. Carol, recomputes her chosen k_C with

$$k_{C_{new}} := k_C - \sum_{i=1}^{w} k_i = k_C - l^{-1}o \pmod{p-1}$$

and the condition is satisfied. Note that l must be chosen relatively prime to $p-1$. If the MIX-channel is secure and at most $w-2$ of the w members of the tiger team collude then there are at least

$$z := \binom{2l}{l}$$

possibilities to map the outputs of the MIX-net to the sender. If the parameter l is chosen large enough such that $z \geq p$ then the (secret) parameters k_i of the tiger team members can't be computed.

How can the second attack be avoided ? One solution is that the clerk is a trusted third party and does not reveal the individual signature parameters s_i. But this assumption is strong and not acceptable for the users. One possible solution seems to be that each of the t signers has to check that the product of any subgroup of the r's is not equal 1. But note that these are

$$\sum_{i=0}^{t-1}\binom{t}{i} = 2^t - 1$$

many and this number increases exponential with the number of signers t. A better solution is that the number of signers or the identity of the signers is appended to the message.

If we sign the original message (which is always the case using message recovery signature schemes) we have the drawback that the expansion rate of the signed message (that is the ratio of the length of the signature parameters and the length of the message) increases. But if we can sign the hashvalue of the message or include the identity of the signers in the signature equation then there is no drawback at all and thus the attack can be countermeasured. We present the resulting schemes in the next sections in detail.

4. The Meta-ElGamal signature scheme

The Meta-ElGamal signature scheme has been proposed in [HMP194] and is based on ideas first published in [HoPe94].

For an ElGamal signature [ElGa84, ElGa85] the trusted authority chooses a large prime p and a generator $\alpha \in \mathbf{Z}_p^*$. p and α are public system parameters and authentically known to all users. The signer *Alice* chooses her secret key $x_A \in \mathbf{Z}_{p-1}^*$ and computes $y_A := \alpha^{x_A} \pmod{p}$. She publishes y_A and keeps x_A secret. These values are constant for all messages to be signed. To sign a message $m \in \mathbf{Z}_{p-1}$ Alice chooses a random number $k \in \mathbf{Z}_{p-1}^*$. She computes $r := \alpha^k \pmod{p}$ and solves the congruence

$$m \equiv x_A r + ks \pmod{p-1} \tag{4}$$

for the parameter s. The triple $(m; r, s)$ is the signed message. It can be verified by checking the congruence

$$\alpha^m \equiv y_A^r r^s \pmod{p}. \tag{5}$$

Instead of signature generation by the equation (4) we can also choose the general equation

$$A \equiv x_A B + kC \pmod{q} \tag{6}$$

with $q \in \Pr, q|(p-1)$, and choose A, B, C as general functions $e, f, g : \mathbf{Z}_q{}^3 \rightarrow \mathbf{Z}_q$ with arguments m, r and s. As $m \in \mathbf{Z}_{p-1}$ we imply that m is reduced modulo q before it is used as an argument but in the following description we omit this for the sake of clearness.

The parameter s should either be used as argument in only one of the three functions or the functions have to be chosen carefully, such that the signature equation can be solved. Also all of the parameters m, r, s have to occur at least once. If two or three functions use exactly the same arguments, then they should be chosen as different operations. The occurrence of the insecure $rs-$ and $ms-$variants [HMP194], where the parameters r and s (m and s) occur exactly in one of the three functions e, f and g together but neither r nor s (m nor s) occurs in one of the two other, should be avoided. All four conditions apply also for equivalent variants, in which the signature equations can be transformed into each other. Furthermore none of the three functions should be equal to zero. To get efficient variants, the functions should be chosen, such that s can be easily extracted (e.g. without inversions). It's also an advantage to choose one of the functions equal to one, to obtain an efficient signature verification. This verification is done by checking the equation

$$\alpha^A \equiv y_A^B r^C \pmod{p}. \tag{7}$$

As there are numerous variants, in the following we will only consider some efficient special cases of permutations, namely to choose A, B, C as a permutation of one of the following five types EG I - EG V, which have been analyzed in detail in [HMP194]:

$$\text{EG I: } (m, r, s), \text{EG II: } (f(m,r), s, 1), \text{EG III: } (f(m,r), g(m,s), 1),$$
$$\text{EG IV: } (f(m,r), g(r,s), 1), \text{EG V: } (f(m,s), g(r,s), 1).$$

The functions $f, g : \mathbf{Z}_q{}^2 \rightarrow \mathbf{Z}_q$ have to be invertible in the argument s to guarantee the solubility of the general signature equation (6) for the signature parameter s.

For every type we get one of the following six permutations of the coefficients, which are enumerated by $No.$ 1 – 6:

$$\begin{array}{lll} 1: (a,b,c) & 2: (a,c,b) & 3: (c,b,a) \\ 4: (c,a,b) & 5: (b,c,a) & 6: (b,a,c) \end{array}$$

For example $(a,b,c) = (m,r,s)$ in *Type* EG I and $(a,b,c) = (f(m,r),s,1)$ in *Type* EG II. Additionally we can generalize the computation of the parameter r by choosing $r' := \alpha^k \pmod p$ and computing $r := d(r', m)$ with a suitable function $d : \mathbf{Z}_p{}^2 \rightarrow \mathbf{Z}_p$. It is also possible to vary the mode of operation that determines the group orders and the length of the parameters [HMP194]:

XL: ElGamal mode with $|p| = |q| = 512$,

L: Schnorr mode [Schn89] with $|p| = 512, |q| = 160$,

M: DSA mode [NIST91] with $|p| = 512, |q| = 160$, r reduced modulo q, and

S: small mode [Schn89, Knob94] with $|p| = 512, |q| = 160$ and a q_1 bit number $h(r)$ $(50 \leq |q_1| \leq 160)$ reduced by any hash function h.

All these generalizations can also be applied to the ElGamal signature scheme with two message blocks (*Type* EG VI – EG X) and the signature scheme with three message blocks (*Type* EG XI) [ElGa84, HMP194]. Combining the described variations we get the Meta-ElGamal signature scheme which can be written as

$$MEG = (Mode.Type.No.\sigma, d, e, f, g).$$

The parameters are chosen in the following way:

- *Mode* $\in$ {XL, L, M, S} gives the mode of operation,
- *No* $\in$ {1, 2, 3, 4, 5, 6} gives the number of the permutation,
- *Type* $\in$ {EG I, EG II, ..., EG XI} gives the type of permutation,
- $d : \mathbf{Z}_p{}^2 \rightarrow \mathbf{Z}_p$ specifies the computation of r,
- $e, f, g : \mathbf{Z}_q{}^3 \rightarrow \mathbf{Z}_q$ invertible in the argument s.

In a simplified manner, we can also describe the Meta-ElGamal scheme by the tuple $(Mode, d, e, f, g)$ but then we loose useful structural information for the security analysis. Therefore we prefer the first notation although it contains redundancy.

5. The Meta-Message recovery scheme

In a similar manner the ideas of the Meta-ElGamal signature scheme were applied to signature schemes giving message recovery [HMP394].

The basic Message recovery scheme

For a signature giving message recovery [NyRu93] the trusted authority chooses large primes p and q, where q is a large integer factor of $p-1$. She also chooses an element $\alpha \in \mathbf{Z}_p^*$ of order q. p, q and α are system parameters and authentically known to all users. The signer Alice chooses x_A and y_A as in the ElGamal scheme. To sign a message $m \in \mathbf{Z}_{p-1}$ satisfying a suitable redundancy scheme, she chooses a random $k \in \mathbf{Z}_q^*$, computes $r' := \alpha^k \pmod{p}$, $r := (r')^{-1}m \pmod{p}$ and solves the congruence $s \equiv k - x_A r \pmod{q}$ for the parameter s. The signature is given by the tuple (r, s). The message can be recovered by computing $m := \alpha^s y_A^r r \pmod{p}$. The correctness of the signature can be checked by the given redundancy scheme.

The Meta-Message recovery scheme

We can apply the Meta-ElGamal scheme to this approach to obtain the Meta-Message recovery scheme [HMP394]. The general signature equation, which has to be solved for the parameter s isof the form

$$A \equiv x_A B + kC \pmod{q} \tag{8}$$

with A, B, C permutations of general functions $e, f, g : \mathbf{Z}_q{}^2 \rightarrow \mathbf{Z}_q$ with arguments r and s. The parameter r can be computed by $r' := \alpha^k \pmod{p}$ and $r := d(m, r')$ with a suitable function $d : \mathbf{Z}_p{}^2 \rightarrow \mathbf{Z}_p$, where $d^{-1}(r, r') = m$ exists. The signature is given by the tuple (r, s) and message recovery can be done by computing

$$m := d^{-1}(r, \alpha^{AC^{-1}} y_A^{-BC^{-1}} \pmod{p}).$$

The verification can be done by checking if m satisfies the redundancy scheme. If we look carefully on the necessary conditions on the functions e, f, g described in section 4, we see that we get the following ten types of permutations:

Type	(A, B, C) permutation of		
MR I	1	r	s
MR II	1	s	$f(r,s)$
MR III	1	r	$f(r,s)$
MR IV	1	$f(r,s)$	$g(r,s)$
MR V	r	r	$f(r,s)$

Type	(A, B, C) permutation of		
MR VI	r	s	$f(r,s)$
MR VII	s	s	$f(r,s)$
MR VIII	r	$f(r,s)$	$g(r,s)$
MR IX	s	$f(r,s)$	$g(r,s)$
MR X	$e(r,s)$	$f(r,s)$	$g(r,s)$

Among these only the types MR I, III and V are solvable for all choices of e, f, g. The most efficient types are the *Type* MR I - IV if we choose the parameter $C = 1$, because we need no inversion for message recovery. In *Type* MR II, IV, VI – X we have to choose suitable functions e, f, g to guarantee the solvability for the parameter s. In *Type* MR IV, we have to choose different functions f, g without homomorphic properties to guarantee the security of the signature scheme. It is argued in [HMP394] that *Mode* L is best suited for message recovery schemes.

We also have message recovery schemes for two and three message blocks, where the functions e, f, g have arguments m_2, r, s. Among these especially the following types, described by the permutation of the coefficients A, B, C will be considered later:

$$\text{MR XI: } (f(m_2,r), s, 1),\ \text{MR XII: } (f(m_2,r), g(m_2,s), 1),$$

$$\text{MR XIII: } (f(m_2,r), g(r,s), 1),\ \text{MR XIV: } (f(m_2,s), g(r,s), 1).$$

6. The Meta-ElGamal multisignature scheme

We now discuss how to adopt the Meta-ElGamal signature scheme to the modified multisignature scheme. We fix the choices of $r_i := \alpha^{k_i} \pmod p$,

$$r' := \prod_{i=1}^{t} r_i \pmod p, r := d(r', m) \text{ and } y := \prod_{i=1}^{t} y_i \pmod p.$$

Furthermore the verification equation $\alpha^A \equiv y^B r^C \pmod p$ should be satisfied where A, B and C are chosen as general functions e, f, g with arguments m, r and s. For the sake of clarity we choose the function d equal to $d(r', m) = r' \pmod p$ and use the *Mode* L. The general case will be discussed briefly at the end of this section. The computation of the signature parameter s_i by the user u_i is done by transforming the congruence

$$A_i \equiv x_i B_i + k_i C_i \pmod q,$$

where A_i, B_i and C_i are choosen as general functions e, f, g with arguments r, m and s_i. Thus we have the following congruence:

$$y^B r^C \equiv \left(\prod_{i=1}^{t} y_i^B\right) r^C \equiv \left(\prod_{i=1}^{t} \alpha^{(A_i - C_i k_i) B_i^{-1} B}\right) r^C \equiv \left(\prod_{i=1}^{t} \alpha^{A_i B_i^{-1} B} \prod_{i=1}^{t} r_i^{-C_i B_i^{-1} B}\right) r^C \pmod p.$$

If C_i and B_i do not depend on s_i, such that $C = C_i$ and $B = B_i$, we get

$$\equiv \left(\prod_{i=1}^{t} \alpha^{A_i} \prod_{i=1}^{t} r_i^{-C}\right) r^C \equiv \alpha^{\sum_{i=1}^{t} A_i} \pmod p.$$

This should be equivalent to α^A to satisfy the verification equation. Hence we get the sufficient conditions for A_i, A, B_i, B, C_i, C that $A \equiv \sum_{i=1}^{t} A_i \pmod q, B = B_i$ and $C = C_i$. The parameter s has to appear in the coefficient A as it must not appear in B and C.

Table 1 gives an overview about the suitable ElGamal variants of the first five types for multisignature schemes. The corresponding types of the Meta-<u>E</u>lGamal <u>M</u>ultisignature

No.	A	B	C	signature	verification
EG I.3	s	r	m	$s \equiv x_A r + km$	$\alpha^s \equiv y_A^r r^m$
EG I.4	s	m	r	$s \equiv x_A m + kr$	$\alpha^s \equiv y_A^m r^r$
EG II.3	s	$f(m,r)$	1	$s \equiv x_A f(m,r) + k$	$\alpha^s \equiv y_A^{f(m,r)} r$
EG II.4	s	1	$f(m,r)$	$s \equiv x_A + kf(m,r)$	$\alpha^s \equiv y_A r^{f(m,r)}$
EG III.3	$g(m,s)$	$f(m,r)$	1	$g(m,s) \equiv x_A f(m,r) + k$	$\alpha^{g(m,s)} \equiv y_A^{f(m,r)} r$
EG III.4	$g(m,s)$	1	$f(m,r)$	$g(m,s) \equiv x_A + kf(m,r)$	$\alpha^{g(m,s)} \equiv y_A r^{f(m,r)}$
EG V.3	$g(r,s)$	1	$f(m,r)$	$g(r,s) \equiv x_A + kf(m,r)$	$\alpha^{g(r,s)} \equiv y_A r^{f(m,r)}$
EG V.4	$g(r,s)$	$f(m,r)$	1	$g(r,s) \equiv x_A f(m,r) + k$	$\alpha^{g(r,s)} \equiv y_A^{f(m,r)} r$

Table 1: Variants of the Meta-ElGamal signature scheme suitable for multisignature

scheme are abbreviated by EM followed by the capital roman number of the corresponding ElGamal variants.

The function f can be chosen arbitrarily as far as the necessary conditions described in section 4 hold, while the function g must satisfy the additional condition that $g(m,s) \equiv \sum_{i=1}^{t} g(m,s_i) \pmod q$ in types EM III.3 and EM III.4 or $g(r,s) \equiv \sum_{i=1}^{t} g(r,s_i) \pmod q$ in types EM V.3 and EM V.4. Note that the computation of s can vary depending on the function g. As a result we can choose an arbitrarily function g which is invertible in the argument s and compute

$$s := g^{-1}\left(\sum_{i=1}^{t} g(m,s_i), m\right) \pmod q \text{ or } s := g^{-1}\left(\sum_{i=1}^{t} g(r,s_i), r\right) \pmod q,$$

where the function g^{-1} is implicitely defined by $g^{-1}(g(a,b),a) = b$.

Now we describe the Meta-ElGamal multisignature scheme in *Mode* L:

1. The trusted third party chooses a primes p and q with $q|(p-1)$ and a generator $\alpha \in \mathbf{Z}_p^*$. These values are published as system parameters.

2. Each user $u_i, i \in [1:t]$ chooses a value $x_i \in \mathbf{Z}_q$ and computes $y_i := \alpha^{x_i} \pmod p$. He keeps x_i secret and sends y_i authentic to the trusted third party. Furthermore he shows that he knows the discrete logarithm of y_i using the zero-knowledge proof due to Chaum, Evertse and van de Graaf. Then the accepted public-keys are published by the trusted third party. Alternatively the user u_i signs a fixed message (which includes the public key y_i) using his secret key x_i. He sends it to the trusted third party who accepts it if y_i has not been used as a public key of any other user so far.

3. If the t users $u_1, \ldots, u_t$ want to sign the message m' every signer u_i computes $m := h(m', ID_1, \ldots, ID_t)$ using a public collision-free hash function $h : \mathbf{Z}_{p-1}^{t+1} \longrightarrow \mathbf{Z}_{p-1}$, where ID_i is the unique identifier of user u_i.

4. Every signer u_i chooses a random $k_i \in \mathbf{Z}_q$, computes $r_i := \alpha^{k_i} \pmod p$ and broadcasts r_i to the other signers and the clerk.

5. Every signer computes $r := \prod_{i=1}^{t} r_i \pmod p$ and the individual signature using a suitable variant of the Meta-ElGamal signature scheme by solving the equation

$$A_i \equiv x_i B + k_i C \pmod q$$

for the signature parameter s_i where the coefficient A_i is chosen as a suitable function with arguments m', s_i and r. Coefficients B and C are chosen as general functions with arguments m' and r. All functions are pre-arranged between all signers. Then he sends s_i to the clerk.

6. The clerk collects the parameters s_i of all signers, checks the individual signatures and computes r and s. For the computation of s he uses an equation such that the congruence $A \equiv \sum_{i=1}^{t} A_i \pmod q$ holds. Then he publishes the message m, the multisignature (r, s) and the identities of the signers.

7. The multisignature can be verified by checking the congruence

$$\alpha^A \equiv y^B r^C \pmod p$$

where $y = \prod_{i=0}^{t} y_i$ and $m = h(m', ID_1, \ldots, ID_t)$.

Another solution for avoiding the second attack described in section 3 is to use signature schemes with two message blocks. This solution might be used if the message itself and not a hash value should be signed. Here we use one message block for the message. The other one contains the hash value of the identities ID_i of all signers. We give a more detailed description of this approach in the next section.

As already mentioned above we can generalize the computation of the parameter r by the function d in the following way [HMP194]: First $r' := \prod_{i=1}^{k} r_i \pmod p$ is computed and then $r := d(r', m)$. If $C^{-1} \pmod q$ is computable (which is always the case in modes L, M and S), we can verify the multisignature by the equation

$$r = d\left(\alpha^{AC^{-1}} y^{-BC^{-1}} \pmod p, m\right).$$

If d is invertible in the argument r', which means that $d^{-1}(r, m) = r'$ exists, we can also verify the multisignature by the equation

$$\alpha^A \equiv y^B \left(d^{-1}(r, m)\right)^C \equiv y^B (r')^C \pmod p.$$

The other modifications for the different modes are straightforward.

Security analysis

First it is interesting to outline how to classify the result of attack and the strength of the attack. Based on the distinctions in [GoMR88] we have

- *A total break:* The attacker can compute the secret key of one of the users,
- *Universal forgery:* The attacker can find signatures of one group of users for nearly all arbitrary messages chosen by him, but doesn't know their secret keys ,
- *Selective forgery:* The attacker can find signatures of one group of users a particular messages chosen by him a priori,
- *Existential forgery:* The attacker can find signatures of one group of users for a message which might be random or nonsencial.

The strength of a attack is determined by the possiblities the attacker possesses: Thus we have key only attacks, message attacks, known-message attacks, generic chosen-message attacks, directed chosen message attacks and adaptive chosen-message attacks [GoMR88]. The first three can be regarded as *weak* and the last four as *strong* attacks. In the multisignature schemes we have several possible attackers: *Outsiders*, who are not registered users, *insiders* who are registered users and can play an active part in the signing process and the *clerk*. Furthermore some of these different attackers might collude.

1. *Outsider attack:* In the known-message scenario the outsider knows the public parameters, some multisignatures and the related individual signatures if he eavesdrops the communication channel or corrupts the clerk. One possiblity he has is a replay attack: He eavesdrops $r_{A,1}$ and $s_{A,1}$ of user Alice from the signature generation and tries to use this for generating a new signature. He puts $r_{A,1}$ into the broadcast channel as the new parameter $r_{A,2}$. As he doesn't know the related parameter $k_{A,2}$ he can't compute $s_{A,2}$. Thus he has to use the eavesdroped value $s_{A,1}$. But this is only the correct value if $m_1' = m_2'$ and $r_1 = r_2$. As the hash function is collision-free the first condition is satisfied in those cases where the same group of signers signs the same message twice. Thus this attack is not successful. Other possibilities might be to forge the individual signature of Alice or the multisignature. Both cases can be seen as the attempt to forge the used variant of the Meta-ElGamal signature scheme. A detailed security analysis for the Meta-ElGamal scheme is described in [HMP294]. Stronger attacks seems to be useless because m' changes if a different group of signers signs the same messsage or different messages are signed. It seems to be hard to combine them to get additional signatures.

2. *Insider attack:* Insiders can try to choose their parameters k_i such that r is chosen suitable, like in the second attack described in section three. If they fix r in advance they must be able to compute discrete logarithms if they don't know at least one of the values r_i. This contradicts the discrete logarithm assumption.

3. *Clerk attack:* The clerk might reveal individual signatures but this doesn't endanger the security of the scheme. If he doesn't compute the multisignature correctly from the individual signatures the verification equation of the multisignature is not satisfied and thus this attack can be discovered by the verifiers.

Performance analysis

For the signature generation each signer has nearly the same computational effort than in the chosen variant of the Meta-ElGamal signature scheme. The clerk has to check each individual signature, for which he needs $2t$ or $3t$ exponentiations depending on the chosen variant. The computation of the multisignature needs low costs if the parameter g is chosen suitable (e.g. as addition or multiplication). The signature size is given by two long integers and the message. For the verification of the multisignature two or three exponentiations are necessary, depending of the used variant. The most efficient variants are corresponding to the most efficient variants of the Meta-ElGamal signature scheme (e.g. EG II.3, EG III.3, EG V.4). Thus the scheme reviewed in section 2 is one of the most efficient multisignature schemes.

7. The Meta-Message recovery multisignature scheme

We can adopt the MMR-signatures to the multisignature scheme. Every signer computes $r_i := \alpha^{k_i} \pmod p$ and $r' := \prod_{i=1}^{t} r_i \pmod p$. The choice of r and y is fixed to

$$r := d(r', m) \text{ and } y := \prod_{i=1}^{t} y_i \pmod p.$$

Furthermore $d^{-1}\left(r, \alpha^{AC^{-1}} y^{-BC^{-1}}\right) = m$ should be satisfied where A, B and C are choosen as a permutation of functions e, f, g. The computation of the signature parameter s_i by the user u_i is done by transforming

$$A_i \equiv x_i B_i + k_i C_i \pmod q,$$

where A_i, B_i and C_i are choosen as general functions with arguments r and s_i. Thus we get

$$d^{-1}\left(r, \alpha^{AC^{-1}} y^{-BC^{-1}} \pmod p\right) = d^{-1}\left(r, \alpha^{AC^{-1}} \left(\prod_{i=1}^{t} y_i^{-BC^{-1}}\right) \pmod p\right)$$

$$= d^{-1}\left(r, \alpha^{AC^{-1}} \left(\prod_{i=1}^{t} \alpha^{-(A_i - C_i k_i) B_i^{-1} BC^{-1}}\right) \pmod p\right)$$

$$= d^{-1}\left(r, \alpha^{AC^{-1}} \left(\prod_{i=1}^{t} \alpha^{-A_i B_i^{-1} BC^{-1}} \prod_{i=1}^{t} r_i^{C_i B_i^{-1} BC^{-1}}\right) \pmod p\right).$$

If C_i and B_i do not depend on the signature parameter s_i, such that $C = C_i$ and $B = B_i$, we get

$$= d^{-1}\left(r, \alpha^{AC^{-1}} \left(\prod_{i=1}^{t} \alpha^{-A_i C^{-1}} \prod_{i=1}^{t} r_i\right) \pmod p\right)$$

$$= d^{-1}\left(r, \alpha^{C^{-1}\left(A - \sum_{i=1}^{t} A_i\right)} \prod_{i=1}^{t} r_i \pmod p\right) = d^{-1}(r, r') = m.$$

Hence we get the sufficient conditions for A_i, A, B_i, B, C_i, C that $A \equiv \sum_{i=1}^{t} A_i \pmod{p-1}$, $B = B_i$ and $C = C_i$. The parameter s has to appear in A because it must not appear in B and C.

Table 2 gives an overview about the variants the first five types of the Meta-Message recovery scheme which are suitable for multisignature schemes, where $d(r', m) = (r')^{-1} \cdot m = r \pmod{p}$ and therefore $d^{-1}(r, r') = r' \cdot r = m \pmod{p}$.

No.	A	$-B$	C	signature	message recovery
MR I.3	s	r	1	$s \equiv -x_A r + k$	$m \equiv \alpha^s y_A^r r$
MR I.4	s	1	r	$s \equiv -x_A + kr$	$m \equiv \alpha^{sr^{-1}} y_A^{r^{-1}} r$
MR III.3	$f(r,s)$	1	r	$f(r,s) \equiv -x_A + kr$	$m \equiv \alpha^{f(r,s)r^{-1}} y_A^{r^{-1}}$
MR III.4	$f(r,s)$	r	1	$f(r,s) \equiv -x_A r + k$	$m \equiv \alpha^{f(r,s)} y_A^r r$

Table 2: Message recovery signature schemes suitable for multisignatures

Note that in message recovery signature schemes the message and not a hash value of the message must be signed if we don't want to append the message to the signature. Furthermore in the schemes with one-message block the data expansion is rather high because the identifiers of the signers must be appended to the message to countermeasure the second attack given in section three. Another solution is to use message recovery signature schemes with two message blocks - one is recovered by the verification equation and the other one is appended to the signature and just contains the identifiers of the signers. We give some efficient variants in table 3, where ID is computed by $ID := h(ID_1, \ldots, ID_t)$ using a public hash function h assuming that the users $u_1, \ldots, u_t$ wants to sign the message m and the function d is chosen as above.

No.	A	$-B$	C	signature	message recovery
MR XI.3	s	$f(r,ID)$	1	$s \equiv -x_A f(r,ID) + k$	$m \equiv \alpha^s y_A^{g(r,ID)} r$
MR XI.4	s	1	$f(r,ID)$	$s \equiv -x_A + kf(r,ID)$	$m \equiv \alpha^{sf(r,ID)^{-1}} y_A^{g(r,ID)^{-1}} r$
MR XIII.3	$f(r,s)$	1	$g(r,ID)$	$f(r,s) \equiv -x_A + kg(r,ID)$	$m \equiv \alpha^{f(r,s)g(r,ID)^{-1}} y_A^{g(r,ID)^{-1}}$
MR XIII.4	$f(r,s)$	$g(r,ID)$	1	$f(r,s) \equiv -x_A g(r,ID) + k$	$m \equiv \alpha^{f(r,s)} y_A^{g(r,ID)} r$

Table 3: Efficient MMR schemes with two messages blocks suitable for multisignatures

A detailed description of the Meta-Message recovery multisignature scheme is similar to the Meta-ElGamal multisignature scheme described above and hence we skip it here.

Security analysis

The security analysis is similar to the security analysis of the Meta-ElGamal multisignature scheme.

Performance analysis

The computational effort for signature generation and verification is similar to the Meta-Message recovery variants. Note that the message needs not be transmitted additionally to the signature parameters, but the message can't be hashed.

8. Conclusion

We presented two attacks on the multisignature scheme proposed by Harn and showed how to countermeasure them. Then we generalized a modification of this scheme to some variants of the Meta-ElGamal signature schemes and suggested the first multisignature schemes giving message recovery. Furthermore we gave sufficient conditions to decide which of the Meta-ElGamal and Meta-Message recovery variants are suitable for multisignatures.

References

[Chau81] D.Chaum, "Untraceable electronic mail return addresses and digital pseudonyms", Communcations of the ACM, Vol. 24 (2), Feb., (1981), pp. 84 - 88.

[ChEG87] D.Chaum, J.-H.Everste, J.van de Graaf, "An improved Protocol for Demonstrating Possession of Discrete Logarithms and some Generalizations", Lecture Notes in Computer Science 304, Advances in Cryptology: Proc. Eurocrypt'87, Berlin: Springer Verlag, (1988), pp. 127 - 143.

[ElGa84] T.ElGamal, "Cryptography and logarithms over finite fields", Stanford University, CA., UMI Order No. DA 8420519, (1984), 119 pages.

[ElGa85] T.ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms", IEEE Transactions on Information Theory, Vol. IT-30, No. 4, July, (1985), pp. 469 - 472.

[GoMR88] S.Goldwasser, S.Micali, R.Rivest, "A digital signature scheme secure against adaptive chosen message attacks", SIAM Journal on Computing, Vol. 17, No. 2, (1988), pp. 281 - 308.

[Har194] L.Harn, "Group-oriented (t,n) threshold digital signature scheme and digital multisignature", IEE Proc.-Comput. Digit. Tech., Vol. 141, No. 5, September, (1994), pp. 307 - 313.

[Har294] L.Harn, "New digital signature scheme based on discrete logarithm", Electronics Letters, Vol. 30, No. 5, (1994), pp. 396 - 398.

[HoPe94] P.Horster, H.Petersen, "Generalized ElGamal-signatures" (in German), Sicherheit in Informationssystemen, Proceedings of SIS '94, Zürich, March 10–11, 1994, Verlag der Fachvereine Zürich, (1994), pp. 89–106.

[HMP194] P.Horster, M.Michels, H.Petersen, "Meta-ElGamal signature schemes", Proc. of the 2nd ACM conference on Computation and Communication security, Fairfax, Virginia, Nov. 2–4, (1994), pp. 96 - 107.

[HMP294] P.Horster, M.Michels, H.Petersen, "Generalized signature schemes for one message block", Proc. Workshop on IT-Security, Vienna, Sept. 22–23, (1994), 16 pages.

[HMP394] P.Horster, M.Michels, H.Petersen, "Meta signature schemes giving message recovery based on the discrete logarithm problem", Proc. Workshop on IT-Security, Vienna, Sept. 22–23, (1994), 12 pages.

[Knob94] H.-J.Knobloch, "A remark on the size of ElGamal-type digital signatures", EISS Report 1/94, University of Karlsruhe, (1994), 5 pages.

[NIST91] National Institute of Standards and Technology, Federal Information Process. Standard, FIPS Pub XX: Digital Signature Standard (DSS), (1991).

[NyRu93] K.Nyberg, R.Rueppel, "A new signature scheme based on the DSA giving message recovery", Proc. 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, Nov. 3–5., (1993), 4 pages.

[NyRu94] K.Nyberg, R.Rueppel, "Message recovery for signature schemes based on the discrete logarithm problem", Pre-proceedings of Eurocrypt '94, (1994), pp. 175 – 190.

[Schn89] C.P.Schnorr, "Efficient identification and signatures for smart cards", Lecture Notes in Computer Science 435, Advances in Cryptology: Proc. Crypto '89, Berlin: Springer Verlag, (1990), pp. 239 - 251.