ON THE SECURITY OF A STRUCTURAL PROVEN SIGNER ORDERING MULTISIGNATURE SCHEME*

Chris J. Mitchell and Namhyun Hur
Mobile VCE Research Group, Information Security Group
Royal Holloway, University of London
Egham, Surrey TW20 0EX, UK
C.Mitchell@rhul.ac.uk, Namhyun.Hur@rhul.ac.uk

Abstract Certain undesirable features are identified in the 'Structural proven signer ordering' multisignature scheme of Kotzanikolaou, Burmester and Chrissikopoulos. This scheme is a modification of a previous multisignature scheme due to Mitomi and Miyaji.

Keywords: mobile agent, digital signature, multisignature, cryptanalysis, security

1. Introduction

The notion of a multisignature scheme was introduced nearly 20 years ago [Itakura and Nakamura, 1983], and a number of schemes have been proposed since that time. The fundamental idea of a multisignature scheme is that it enables a number of users to collectively create a digital signature on a document (using their own private keys). Typically, all users will sign the same document, and either the order in which they sign will be fixed or, if it is not fixed, then the verifier will not be able to determine in which order the various users signed the document.

For further details on such multisignature techniques, and also on the ElGamal signature scheme on which the cryptosystems described in this paper are based, see, for example, [Menezes et al., 1997].

*The work reported in this paper has formed part of the Software Based Systems work area of the Core 2 Research Programme of the Virtual Centre of Excellence in Mobile & Personal Communications, Mobile VCE, www.mobilevce.com, whose funding support, including that of EPSRC, is gratefully acknowledged. More detailed technical reports on this research are available to Industrial Members of Mobile VCE.

1.1. Mitomi-Miyaji multisignatures

Recent papers [Mitomi and Miyaji, 2000, Mitomi and Miyaji, 2001] extend the notion of a multisignature. They provide a model for a multisignature scheme that allows three key properties:

- *message flexibility*, i.e., each party can sign a different document,
- *order flexibility*, i.e., the order in which the various parties create their contribution to the multisignature is not fixed, and
- *order verifiability*, i.e., the order in which the various parties created their contribution to the multisignature can be verified by the verifier of the multisignature.

Mitomi and Miyaji also propose two different multisignature schemes fitting this model, one discrete logarithm based and the other RSA based.

1.2. Multisignatures for mobile agents

In [Kotzanikolaou et al., 2001], the application of Mitomi-Miyaji multisignatures to a mobile agent environment is considered. Specifically, mobile agents (essentially autonomous pieces of code) may visit a number of host platforms, and may wish to collectively sign a message, e.g. to commit to a transaction on behalf of the original sponsor of the agents. Each agent will be equipped with its own (multi)signature private key.

The reason to employ such a model is that single agents may not be trusted to complete a transaction on behalf of a remote sponsor, since their operation may be interfered with by the platform on which they run. In general, there are a number of ways in which the threat posed by a small number of malicious platforms can be reduced. One such approach is to send multiple copies of a transaction agent to a number of platforms, and require that a certain number of copies of the agent (running on different platforms) all consent before the transaction is completed. Each copy of the agent is equipped with a distinct signature key pair (thus preventing an agent on one platform masquerading as an agent executing on a different platform). Of course such an approach requires some co-ordination amongst the various platforms involved, but this is not an issue we consider further here.

A variant of the above approach motivates the particular application of multisignatures we consider here. The model discussed in [Kotzanikolaou et al., 2001] involves a series of agents: $U_1, U_2, \ldots, U_n$ each contributing to a multisignature in turn. Each agent U_i adds its own message string m_i to the evolving multisignature, and thus user U_i actually contributes to a multisignature on a sequence of messages $m_1, m_2, \ldots, m_i$.

We suppose that the recipient of the multisignature will only accept it if a minimum number of distinct agents have contributed to the signature, and that all the agent messages m_i are 'consistent' in some application-specific way.

In this context, [Kotzanikolaou et al., 2001] identify a potential problem with use of Mitomi-Miyaji multisignatures. Specifically, a malicious user can delete one or more of the most recent agent contributions from a multisignature (Kotzanikolaou et al. call this an *exclude* attack). Kotzanikolaou et al. propose two different ways of addressing this problem.

- The first approach, described in Section 3.4 of [Kotzanikolaou et al., 2001], is called a 'simple solution'. It requires signing agent U_j to include in message m_j the identity of U_{j+1}, the agent which U_j selects to be the next entity to contribute to the multisignature. This clearly prevents a malicious party from 'winding back' a multisignature. No changes to the Mitomi-Miyaji schemes are required.

- The second method, described in Section 3.5 of [Kotzanikolaou et al., 2001], is called 'structural proven signer ordering'. This solution actually involves a minor modification to the discrete logarithm based Mitomi-Miyaji scheme. The multisignature computation performed by U_j is modified to include the value of the public key of the next party to the multisignature, namely U_{j+1}. This is designed to achieve the same objective as the simple solution.

  Unfortunately, as we describe below, it is precisely this small modification that enables the manipulation of multisignatures in certain special circumstances. The main conclusion of this paper is therefore that the 'simple solution' is probably preferable.

Specifically, in the remainder of this paper we describe two undesirable features of the structural proven signer modification to the Mitomi-Miyaji discrete logarithm based multisignature scheme.

1.3. Notation and assumptions

We use the notation of [Kotzanikolaou et al., 2001]. Specifically, we suppose that a multisignature is being computed by a series of signers $U_1, U_2, \ldots, U_j$. The part multisignature output by user U_j consists of two sequences of values, namely the messages $m_1, m_2, \ldots, m_j$ (where m_i is chosen by U_i, $1 \leq i \leq j$), and the multisignature components $s_1, s_2, \ldots, s_j$ (where s_i is computed by U_i, $1 \leq i \leq j$), together with the single value r_j.

As in the scheme described in Section 3.5 of [Kotzanikolaou et al., 2001], we suppose that p and g are universally agreed domain parameters, where p is a large prime satisfying $p = 2q + 1$, q is also prime, and g $(1 < g < p)$ has multiplicative order q modulo p.

2. A (partial) message manipulation attack

Suppose a malicious user has succeeded in obtaining iq as its public key, for some integer i. Of course, in general, the malicious user will not know the private key for this public key, i.e. the malicious user will not know a value x for which $g^x \bmod p = iq$. However, this does not prevent at least a partial attack, as we now describe.

2.1. The partial attack

Suppose that a multisignature is being constructed (using the method in Section 3.5 of [Kotzanikolaou et al., 2001]) by a series of signers $U_1, U_2, \ldots, U_j$, and that the next signer (U_{j+1}) is the malicious user; hence U_{j+1} has $y_{j+1} = iq$ as its public key. For convenience we also suppose that $j > 1$, although the attack will work in almost exactly the same way if $j = 1$.

Using the notation of [Kotzanikolaou et al., 2001], U_j will compute

$$\begin{aligned} R_j &= g^{k_j} \bmod p, \\ r_j &= (h(m_j||\mathrm{ID}_j) \cdot r_{j-1})^{-1} \cdot R_j \bmod q, \quad \text{and} \\ s_j &= (x_j r_j + y_{j+1}) \cdot k_j^{-1} \bmod q \end{aligned}$$

where x_j is the private key of U_j, h is a hash-function, and y_{j+1} is the public key of user U_{j+1}. Hence, since we know that $y_{j+1} \bmod q = 0$, we have

$$s_j = x_j r_j k_j^{-1} \bmod q.$$

User U_j then sends r_j, s_j and m_j to U_{j+1} (together with various other values not of relevance here).

User U_{j+1} can now change the message m_j which user U_j signed. Specifically, suppose user U_{j+1} wishes to make it look as though user U_j signed message $m'_j \neq m_j$. User U_{j+1} first computes $h(m'_j||\mathrm{ID}_j)$ and then computes

$$r'_j = r_j \cdot h(m_j||\mathrm{ID}_j) \cdot (h(m'_j||\mathrm{ID}_j))^{-1} \bmod q.$$

This requires no special knowledge. However, the fact that $y_{j+1} \bmod q = 0$ enables U_{j+1} to compute the 'matching' value s'_j using

$$s'_j = s_j r_j^{-1} r'_j \bmod q = x_j r'_j k_j^{-1} \bmod q = (x_j r'_j + y_{j+1}) \cdot k_j^{-1} \bmod q.$$

These new values r'_j and s'_j can now be used to replace r_j and s_j in the (partial) multisignature, at the same time that m'_j replaces m_j.

2.2. Completing the attack

Whether or not the process described above is a serious attack depends on whether or not U_{j+1} is in a position to complete the modified multisignature. This depends on whether U_{j+1} possesses the private key x_{j+1} corresponding to the public key $y_{j+1} = iq$. In general this appears to be difficult to arrange.

However, there is one specific case where it is possible for a malicious user to calculate the private key corresponding to a public key congruent to zero modulo q. Suppose, as is often described, the domain parameters p and g are selected as follows.

1 p is chosen so that $q = (p-1)/2$ is prime, and thus precisely $q-1$ of the $p-1$ non-zero elements modulo p, i.e. approximately 50%, will be primitive (see, for example, Section 4.6.1 of [Menezes et al., 1997]).

2 A primitive element modulo p is chosen; call this value e.

3 g is set equal to e^2, guaranteeing that g has order q.

Suppose moreover that $e = 2$. This is not unlikely to be the case; heuristically we expect 2 to be primitive roughly half the time, since roughly half the non-zero elements are primitive, and 2 is typically the first value chosen in a search for a primitive element. In such a case we have $g = 2^2 \bmod p = 4$.

Next observe that $2^q \bmod p = p - 1 = 2q$, and hence $2^{q-1} \bmod p = q$. Thus, $g^{(q-1)/2} \bmod p = 2^{q-1} \bmod p = q$. That is, the private key corresponding to the public key q is simply $(q-1)/2$. Hence, in this special case, if the malicious user chooses his/her public key to be q, then he/she will know his/her own private key, and hence would be able to complete the forged partial multisignature. This represents a serious compromise of the security of the scheme.

Of course, if this particular special case is avoided then the partial signature cannot be completed and the 'partial attack' is simply a (probably unexploitable) questionable property of the scheme.

Finally note that there is one other way in which the above situation can arise. Suppose that, after selecting p (and hence q), g is found by successively examining values 2, 3, 4, and so on, until an element of order q is found. This is a reasonable approach, since small values of g have implementation advantages. Suppose also that 2 and 3 are primitive (and hence are not suitable) — as previously, using heuristic arguments we expect this to be true roughly 25% of the time. Then 4 will have order q and will be selected — exactly the same situation now arises.

3. A destination manipulation attack

We show how three different users can conspire to manipulate a contribution to a multisignature made by an honest user.

Suppose that a multisignature is being constructed (using the method in Section 3.5 of [Kotzanikolaou et al., 2001]) by a series of signers $U_1, U_2, \ldots, U_j$, where $j > 2$.

Then, using the notation of [Kotzanikolaou et al., 2001], U_{j-1} will compute

$$\begin{aligned} R_{j-1} &= g^{k_{j-1}} \bmod p, \\ r_{j-1} &= (h(m_{j-1}||\mathrm{ID}_{j-1}) \cdot r_{j-2})^{-1} \cdot R_{j-1} \bmod q, \quad \text{and} \\ s_{j-1} &= (x_{j-1}r_{j-1} + y_j) \cdot k_{j-1}^{-1} \bmod q \end{aligned}$$

where x_{j-1} is the private key of U_{j-1}, h is a hash-function, and y_j is the public key of user U_j. User U_{j-1} then sends r_{j-1}, s_{j-1} and m_{j-1} to U_j (together with various other values not of relevance here).

Similarly, U_j will compute

$$\begin{aligned} R_j &= g^{k_j} \bmod p, \\ r_j &= (h(m_j||\mathrm{ID}_j) \cdot r_{j-1})^{-1} \cdot R_j \bmod q, \quad \text{and} \\ s_j &= (x_j r_j + y_{j+1}) \cdot k_j^{-1} \bmod q \end{aligned}$$

where x_j is the private key of U_j and y_{j+1} is the public key of user U_{j+1}. User U_j then sends r_j, s_j and m_j to U_{j+1} (together with various other values not of relevance here).

We now show how a collaboration of three users, namely U_{j-1}, U_{j+1} and a third user which we denote by U'_{j+1}, can modify the multisignature contribution of user U_j to make it look as though the next user specified by U_j was U'_{j+1} and not U_{j+1}. The modifications required are as follows.

First, when computing the original values of R_{j-1}, r_{j-1} and s_{j-1}, user U_{j-1} must choose k_{j-1} equal to x'_{j+1}, where x'_{j+1} is the private key of user U'_{j+1} (we also denote the private key of user U_{j+1} by x_{j+1}). Hence

$$R_{j-1} = g^{k_{j-1}} \bmod p = g^{x'_{j+1}} \bmod p = y'_{j+1}.$$

Second, the values R_{j-1}, r_{j-1} and s_{j-1} are replaced with new values R'_{j-1}, r'_{j-1} and s'_{j-1} computed using a new ‘random value’ k'_{j-1}, where $k'_{j-1} = x_{j+1}$, the private key of user U_{j+1}.

The replacement values are now computed as follows:

$$\begin{aligned} R'_{j-1} &= g^{k'_{j-1}} \bmod p = g^{x_{j+1}} \bmod p = y_{j+1}, \\ r'_{j-1} &= r_{j-1} \cdot (y'_{j+1} \bmod q)^{-1} \cdot (y_{j+1} \bmod q) \bmod q \\ &= r_{j-1} \cdot (R_{j-1} \bmod q)^{-1} \cdot (R'_{j-1} \bmod q) \bmod q \\ &= (h(m_{j-1}||\mathrm{ID}_{j-1}) \cdot r_{j-2})^{-1} \cdot R'_{j-1} \bmod q, \\ s'_{j-1} &= (x_{j-1}r'_{j-1} + y_j) \cdot (k'_{j-1})^{-1} \bmod q \\ &= (x_{j-1}r'_{j-1} + y_j) \cdot (x_{j+1})^{-1} \bmod q. \end{aligned}$$

(Note that computing these replacement values is simple since U_{j-1} is a member of the conspiracy).

Replacement values are also computed for R_j, r_j and s_j as follows, this time *without* the co-operation of user U_j:

$$\begin{aligned} R'_j &= R_j, \quad (k_j \text{ is thus as before}), \\ r'_j &= r_j \cdot (r'_{j-1})^{-1} \cdot r_{j-1} \bmod q \\ &= (h(m_j||\mathrm{ID}_j) \cdot r'_{j-1})^{-1} \cdot R'_j \bmod q, \\ s'_j &= s_j \cdot r'_j \cdot (r_j)^{-1} \bmod q. \end{aligned}$$

It remains to show that s'_j has the required properties. Observe that

$$\begin{aligned} s'_j &= s_j \cdot r'_j \cdot (r_j)^{-1} \bmod q, \\ &= (x_j r_j + y_{j+1}) \cdot k_j^{-1} \cdot r'_j \cdot (r_j)^{-1} \bmod q, \quad (\text{by definition of } s_j), \\ &= (x_j r'_j + y_{j+1} \cdot r'_j \cdot (r_j)^{-1}) \cdot k_j^{-1} \bmod q, \\ &= (x_j r'_j + y_{j+1} \cdot r_{j-1} \cdot (r'_{j-1})^{-1}) \cdot k_j^{-1} \bmod q, \quad (\text{by definition of } r'_j), \\ &= (x_j r'_j + y'_{j+1}) \cdot k_j^{-1} \bmod q \quad (\text{by definition of } r'_{j-1}). \end{aligned}$$

This completes the demonstration, since it is clear that s'_j identifies U'_{j+1} as the next participant in the multisignature instead of U_{j+1}.

4. Analysis

Observe that, in most circumstances, the (partial) forgery described in Section 2 cannot be completed to a full multisignature. Hence its impact is very limited. Moreover, if users are required to prove possession of their private key before their public key is certified (or otherwise distributed), as is now deemed 'good practice', then in most cases the partial attack is prevented. However, the existence of such a partial attack (which can be extended to a full attack in certain special cases) is nevertheless of concern.

In addition, whilst the forgery described in Section 3 works, and contravenes the required properties of the scheme, it does so in a relatively weak way (given the need for three parties to collaborate to make a small change to the victim's signature). That is, it is hard to see how this attack could be exploited to damage real users of the scheme. However, the existence of such an attack does raise serious questions about the usability of the scheme.

It would therefore appear wise to use the 'simple solution for proven signer ordering' solution, as proposed in Section 3.4 of [Kotzanikolaou et al., 2001], as opposed to the 'structural proven signer ordering' scheme given in Section 3.5 of [Kotzanikolaou et al., 2001]. The use of this former solution has the advantage that it does not change the Mitomi-Miyaji scheme, which has a proof of security, and it is also applicable to any appropriate multisignature scheme.

References

[Itakura and Nakamura, 1983] Itakura, K. and Nakamura, K. (1983). A public-key cryptosystem suitable for digital multisignatures. *NEC J. Res. Dev.*, **71**.

[Kotzanikolaou et al., 2001] Kotzanikolaou, P., Burmester, M., and Chrissikopoulos, V. (2001). Dynamic multi-signatures for secure autonomous agents. In Tjoa, A. and Wagner, R., editors, *DEXA Workshops — Proceedings 12th International Workshop on Database and Expert Systems Applications (DEXA 2001), 3-7 September 2001, Munich, Germany*, pages 587–591. IEEE Computer Society.

[Menezes et al., 1997] Menezes, A., van Oorschot, P., and Vanstone, S. (1997). *Handbook of Applied Cryptography.* CRC Press, Boca Raton.

[Mitomi and Miyaji, 2000] Mitomi, S. and Miyaji, A. (2000). A multisignature scheme with message flexibility, order flexibility and order verifiability. In Dawson, E., Clark, A., and Boyd, C., editors, *Information Security and Privacy: Proceedings of the 5th Australasian Conference — ACISP 2000*, number 1841 in Lecture Notes in Computer Science, pages 298–312. Springer-Verlag, Berlin.

[Mitomi and Miyaji, 2001] Mitomi, S. and Miyaji, A. (2001). A general model of multisignature schemes with message flexibility, order flexibility and order verifiability. *IEICE Trans. Fundamentals*, **E84–A**:2488–2499.