A multisignature scheme based on the SDLP and on the IFP

R. Durán Díaz[1], L. Hernández Encinas[2], and J. Muñoz Masqué[2]

[1] Universidad de Alcalá, 28871-Alcalá de Henares, Spain
`raul.duran@uah.es`
[2] Instituto de Física Aplicada, CSIC, C/ Serrano 144, 28006-Madrid, Spain
`{luis, jaime}@iec.csic.es`

Abstract. Multisignature schemes are digital signature schemes that permit one to determine a unique signature for a given message, depending on the signatures of all the members of a specific group. In this work, we present a new semi-short multisignature scheme based on the Subgroup Discrete Logarithm Problem (SDLP) and on the Integer Factorization Problem (IFP). The scheme can be carried out in an on- and off-line basis, is efficient, and the bitlength of the multisignature does not depend on the number of signers.

Key words: Digital signature, Multisignature, Public key cryptography

1 Introduction

There are currently different methods and algorithms to perform, in a safe way, digital signatures. Most of these protocols are based on Public Key Cryptography [1]. The main feature of this kind of cryptography is that each individual has two keys, one public key and one private key. Additionally, to make more efficient the procedures of digital signatures and their electronic transmission, hash functions are used [2]. These functions are publicly known and allow signing a digest of the original document instead of the whole document.

Multisignature schemes are protocols of digital signature whereby a group of users, $G = \{U_1, \ldots, U_t\}$, signs a document such that the signature is valid if and only if all members of the group take part in the protocol and the signature verifies a specific condition of validity. These schemes have application in settings such us, for example, corporate scenarios for signing contracts between companies, the government and public administrations, agreements between different organization, etc. The easiest way to carry out a multisignature for a message is to consider as such signature the list formed by all the partial signatures of each one of the signers. However, this signature is not practical since its length is proportional to the number of signers [3, 4].

In general, most of the multisignature protocols are performed as follows:

1. The signer U_1 signs the original message by using the signer private key.

2. Each one of the following signers, in an ordered way, signs the document, already signed by the one who is previous in the group.
3. The last member of G, U_t, signs the signed document that the previous signer has sent to him and sends to the verifier the original message and the multisignature calculated by the group of signers.

The verifier performs the verification of the multisignature by checking each one of the partial signatures of the group of signers, following the protocol and keeping the order in which they were signed.

The first multisignature scheme was proposed in [5], where a modification of the RSA cryptosystem was performed in such a way that the module considered was the product of three primes instead of just two. In [6] a scheme was proposed where the signature length is similar to the length of a simple signature and shorter than the signature obtained from the scheme proposed in [5]. This proposal can be used only if the cryptosystem is bijective. Other proposals based on the RSA cryptosystems have been proposed [7–11].

Regarding multisignature schemes based on the discrete logarithm problem, in [12] the group of signers must cooperate to sign the message and send the signature to a given group of verifiers. Only the union of all verifiers is able to validate the multisignature. Additionally, the signers must use not only their own private keys, but also the public key of all the verifiers. However, this scheme has some weaknesses [13, 14]. The scheme proposed in [15] allows to perform a multisignature if the verifiers of the signature belong to a previously specified group.This scheme has some weaknesses as well [16, 17].

In [18] a multisignature scheme for a generic model of public key is presented. The model requires some properties: Each one of the signers must have a certified public key with its corresponding private key, which must be generated by the signer himself. The signers must interact in a given number of rounds. In each round each signer receives a message, performs several calculations and sends another message to the next signer. It must be computationally infeasible to forge a multisignature if there exists one honest signer.

Our multisignature scheme has the property and advantage that each signer has his own private key, but all of them share the same public key. In this sense, the new scheme does not match exactly the model proposed in [18] since the procedure is carried out in just one round in which all the signers participate. Moreover, each signer does not need to have his own certified pair of keys (public and private). In fact, in the protocol all the signers share the same public key, but each one has his own private key. This fact simplifies and spares some of the problems related to the computational effort for computation, bandwidth, and, therefore, the overall efficiency of the proposed protocol.

Our proposal verifies several properties: It is secure, efficient, independent of the number of signers, the signature is determined by all the signers in any previously given order, allows adding new signers, and the verification procedure does require the verification of the partial signature of each member of G.

2 A multisignature scheme based on SDLP and IFP

We propose a new multisignature scheme whereby each member of a given group, G, signs a document making use of his private key. The verifier of the signature checks whether the signature corresponds to the multisignature of the group, by using the public key that all the members of the group share [19].

We suppose that $G = \{U_1, U_2, \ldots, U_t\}$ is the group of signer and $\mathcal{T}$ is the Trusted Third Party which computes its own private key, the unique public key associated to all private keys, as well as helps the members of G to generate their private key.

2.1 Key generation

First of all, $\mathcal{T}$ generates its own private key:

1. $\mathcal{T}$ chooses two large primes p and q such that
$$p = u_1 \cdot r \cdot p_1 + 1, \quad q = u_2 \cdot r \cdot q_1 + 1,$$
with r, p_1, q_1 primes, $u_1, u_2 \in \mathbb{Z}$, with $\gcd(u_1, u_2) = 2$, *i.e.*, $u_1 = 2v_1$, $v_2 = 2v_2$, and $\gcd(v_1, v_2) = 1$. To guarantee the security of the scheme, the bitlength of r is chosen so that the Discrete Logarithm Problem in a Subgroup of $\mathbb{Z}_n^*$, of order r, be computationally infeasible. Although the factors of n are of a particular form, they can be efficiently generated and to our knowledge there is no known efficient algorithm to factorize n ([20], [21]).
2. $\mathcal{T}$ computes
$$\begin{aligned} n &= p \cdot q, \\ \phi(n) &= (p-1)(q-1) = u_1 \cdot u_2 \cdot r^2 \cdot p_1 \cdot q_1, \\ \lambda(n) &= \operatorname{lcm}(p-1, q-1) = \frac{\phi(n)}{\gcd(p-1, q-1)} = 2v_1 \cdot v_2 \cdot r \cdot p_1 \cdot q_1, \end{aligned}$$
where $\phi(n)$ is the Euler function and $\lambda(n)$ is the Carmichael function.
3. Next, $\mathcal{T}$ selects an element $\alpha \in \mathbb{Z}_n^*$ of order r modulo n, verifying
$$\gcd(\alpha, \phi(n)) = \gcd(\alpha, u_1 \cdot u_2 \cdot r^2 \cdot p_1 \cdot q_1) = 1.$$
The element α can be efficiently computed due to the fact that $\mathcal{T}$ knows the factorization of n, $\phi(n)$, and $\lambda(n)$ [21, Lemma 3.1]. We denote by S_r the multiplicative subgroup of $\mathbb{Z}_n^*$ generated by α.
4. $\mathcal{T}$ generates a secret random number $s \in \mathbb{Z}_r^*$ and computes
$$\beta \equiv \alpha^s \pmod{n}. \tag{1}$$
5. The values (α, r, β, n) are made public; whereas $\mathcal{T}$ keeps secret (p, q, s).

Remark that breaking the key generation protocol amounts to solving the Integer factorization Problem (IFP). Moreover, to determine s from β in the expression (1) the Subgroup Discrete Logarithm Problem (SDLP) must be solved.

Before generating the private key of each signer, $\mathcal{T}$ generates its private key and the shared public key as follows:

1. $\mathcal{T}$ determines its private key by generating four random integer numbers $a_0, b_0, c_0, d_0 \in \mathbb{Z}_r^*$.
2. $\mathcal{T}$ obtains the common public key by computing

$$\begin{aligned} P &\equiv \alpha^{a_0} \cdot \beta^{b_0} \pmod{n} \equiv \alpha^{a_0 + s \cdot b_0} \equiv \alpha^h, \\ Q &\equiv \alpha^{c_0} \cdot \beta^{d_0} \pmod{n} \equiv \alpha^{c_0 + s \cdot d_0} \equiv \alpha^k. \end{aligned}$$

   where $h \equiv (a_0 + s \cdot b_0) \pmod{r}$ and $k \equiv (c_0 + s \cdot d_0) \pmod{r}$.

For avoiding $\mathcal{T}$ can impersonate any signer of G, an interactive session between each user U_i and $\mathcal{T}$ is developed to compute U_i's private key, $i = 1, \ldots, t$:

1. U_i generates two secret integers $b_i, d_i \in \mathbb{Z}_r$ at random and sends the values of $\alpha^{b_i}, \alpha^{d_i}$ to $\mathcal{T}$ in a secure way, in order to protect both secret integers. Note that $\mathcal{T}$ can determine A_i and C_i since it knows h, k, α^{b_i}, and α^{d_i}, but it cannot compute a_i, c_i because it cannot solve the SDLP. In short, each party gets access to only 2 out of the 4 key parameters.
2. $\mathcal{T}$ computes

$$\begin{aligned} A_i &\equiv \alpha^h \cdot (\alpha^{b_i})^{-s} \pmod{n} \equiv \alpha^{a_i}, \\ C_i &\equiv \alpha^k \cdot (\alpha^{d_i})^{-s} \pmod{n} \equiv \alpha^{c_i}. \end{aligned}$$

   Then $\mathcal{T}$ sends to U_i the values of A_i, C_i by using a secure channel.
3. The private key of U_i is the set (b_i, d_i, A_i, C_i). Remark that for U_i it is also impossible to compute the values of a_i and c_i.

2.2 Key verification

To verify the correctness of $\mathcal{T}$'s key, each signer, $U_i \in G$, $i = 1, \ldots, t$, tests if

$$\alpha \not\equiv 1 \pmod{n}, \quad \alpha^r \equiv 1 \pmod{n}.$$

Moreover, each signer must verify that his private key corresponds to the public key (P, Q) by checking the correctness of the following expressions:

$$P \equiv A_i \cdot \beta^{b_i} \pmod{n}, \quad Q \equiv C_i \cdot \beta^{d_i} \pmod{n}.$$

In fact, we have:

$$\begin{aligned} A_i \cdot \beta^{b_i} \pmod{n} &\equiv \alpha^{a_i} \cdot \beta^{b_i} \equiv \alpha^{a_i + s \cdot b_i} \equiv \alpha^h \equiv P, \\ C_i \cdot \beta^{d_i} \pmod{n} &\equiv \alpha^{c_i} \cdot \beta^{d_i} \equiv \alpha^{c_i + s \cdot d_i} \equiv \alpha^k \equiv Q. \end{aligned}$$

2.3 Signing a message

We will present a protocol to determine a multisignature of the group G for a given message M, where only the signers participate.

We suppose a secure hash function, $\mathfrak{h}$, has been selected (for example, one of the SHA-2 family) with $\mathfrak{h}(M) = m$. Moreover, it is assumed that the set of signers has been ordered, due to the fact that each signer will sign the signature determined by the previous signer.

The process is as follows: Each signer verifies the *partial signature* determined by the previous signer, computes his own signature by using the received signature, and sends the new partial signature to the next signer.

1. The first signer, U_1, computes his partial signature for the message M by using his private key, (b_1, d_1, A_1, C_1), and $m = \mathfrak{h}(M)$:

   $$F_1 \equiv A_1 \cdot C_1^m \pmod n,$$
   $$g_1 \equiv b_1 + m \cdot d_1 \pmod r$$

   and sends (F_1, g_1) to the second signer, U_2.
2. The second signer, U_2, verifies U_1's signature checking if

   $$P \cdot Q^m \equiv F_1 \cdot \beta^{g_1} \pmod n.$$

   U_2 computes his partial signature for the message:

   $$F_2 \equiv F_1 \cdot A_2 \cdot C_2^m \pmod n \equiv \alpha^{a_1+a_2+m(c_1+c_2)},$$
   $$g_2 \equiv g_1 + b_2 + m \cdot d_2 \pmod r \equiv b_1 + b_2 + m(d_1 + d_2).$$

   U_2 sends (F_2, g_2) as his partial signature to the third signer.

...

i. The signer U_i receives the U_{i-1}'s partial signature (F_{i-1}, g_{i-1}) and then verifies this partial signature checking if

   $$P^{i-1} \cdot Q^{(i-1)\cdot m} \equiv F_{i-1} \cdot \beta^{g_{i-1}} \pmod n.$$

   U_i computes his partial signature:

   $$F_i \equiv F_{i-1} \cdot A_i \cdot C_i^m \pmod n \equiv \alpha^{a_1+\cdots+a_i+m(c_1+\cdots+c_i)},$$
   $$g_i \equiv g_{i-1} + b_i + m \cdot d_i \pmod r \equiv b_1 + \cdots + b_i + m(d_1 + \cdots + d_i).$$

   U_i sends (F_i, g_i) to the next signer.

...

t. The last signer in the group, U_t, receives the U_{t-1}'s partial signature and verifies that signature testing if

   $$P^{t-1} \cdot Q^{(t-1)\cdot m} \equiv F_{t-1} \cdot \beta^{g_{t-1}} \pmod n.$$

   U_t computes his partial signature for the message:

   $$F_t \equiv F_{t-1} \cdot A_t \cdot C_t^m \pmod n \equiv \alpha^{a_1+\cdots+a_t+m(c_1+\cdots+c_t)},$$
   $$g_t \equiv g_{t-1} + b_t + d_t \cdot m \pmod r \equiv b_1 + \cdots + b_t + m(d_1 + \cdots + d_t).$$

   U_t makes public the multisignature for M: $(F, g) = (F_t, g_t)$.

The verification of each partial signature carried out by each signer (but the first one) is necessary in order to avoid that a signer signs a non-valid message. Moreover, the verification of the U_i's partial signature is correct because it is

$$\begin{aligned} F_i \cdot \beta^{g_i} \pmod n &\equiv \alpha^{a_1+\cdots+a_i+m(c_1+\cdots+c_i)}\beta^{b_1+\cdots+b_i+m(d_1+\cdots+d_i)} \\ &\equiv \alpha^{a_1+\cdots+a_i}(\alpha^{c_1+\cdots+c_i})^m\beta^{b_1+\cdots+b_i}(\beta^{d_1+\cdots+d_i})^m \\ &\equiv \prod_{j=1}^{i}\alpha^{a_j}\cdot\beta^{b_j}\left(\alpha^{c_j}\cdot\beta^{d_j}\right)^m \equiv \prod_{j=1}^{i} P\cdot Q^m = P^i\cdot Q^{i\cdot m}. \end{aligned}$$

2.4 Verifying the multisignature

Let (F, g) be the multisignature for a message M computed by the group of t signers, G. In order to verify such signature, a verifier must to check if

$$P^t\cdot Q^{t\cdot m} \equiv F\cdot\beta^g \pmod n. \tag{2}$$

This verification equation is correct as

$$\begin{aligned} F\cdot\beta^g \pmod n &\equiv \alpha^{a_1+\cdots+a_t+m(c_1+\cdots+c_t)}\beta^{b_1+\cdots+b_t+m(d_1+\cdots+d_t)} \\ &\equiv \prod_{j=1}^{t}\alpha^{a_j}\cdot\beta^{b_j}\left(\alpha^{c_j}\cdot\beta^{d_j}\right)^m \equiv \prod_{j=1}^{t} P\cdot Q^m = P^t\cdot Q^{t\cdot m}. \end{aligned}$$

2.5 Properties and Security analysis

The proposed multisignature scheme has the following properties:

1. The scheme has a fixed size, *i.e.*, it does not depend on the number of signers.
2. The multisignature is a semi-short signature in the sense that the pair (F, g) is composed by two elements belonging to $\mathbb{Z}_n^*$ and to $\mathbb{Z}_r^*$, respectively.
3. The multisignature is efficient as all computations require polynomial time.
4. It is possible to include new signers in the group G without re-execution of the protocol by the rest of the signers. It is possible to place the new signers at the end of the signer group so that each one of them follows the protocol by computing his partial signature from the previously computed multisignature.
5. The multisignature verification process is easy and efficient.

The proposed multisignature scheme is secure since to break the proposed scheme an attacker needs to solve three difficult problems: IFP, DLP, and SDLP. Hence, a signer knowing only his private key cannot determine neither $\mathcal{T}$'s private key nor its secret value s.

In the scheme it is impossible for two signers to compute a forged signature because each signer verifies the signatures of all the previous signers.

Moreover, two or more signers could try to conspire with the goal of obtaining the secret value s of $\mathcal{T}$, and then computing new private keys.

In this attack, if the signers U_i and U_j, $j > i$, share their signatures (F_i, g_i) and (F_j, g_j), they know that the following holds

$$\begin{aligned} F_i \cdot \beta^{g_i} &\equiv F_j \cdot \beta^{g_j} \pmod{n}, \\ A_i \cdot \beta^{b_i} \cdot C_i^m \cdot \beta^{md_i} &\equiv A_j \cdot \beta^{b_j} \cdot C_j^m \cdot \beta^{md_j} \pmod{n}, \\ \alpha^{a_i + s\cdot b_i + m\cdot c_i + s\cdot m\cdot d_i} &\equiv \alpha^{a_j + s\cdot b_j + m\cdot c_j + s\cdot m\cdot d_j} \pmod{n}. \end{aligned}$$

Then, they can suppose that the exponents verify the following equations:

$$\begin{gathered} a_i + s\cdot b_i + m\cdot c_i + s\cdot m\cdot d_i \equiv a_j + s\cdot b_j + m\cdot c_j + s\cdot m\cdot d_j \pmod{r}, \\ a_i - a_j + m(c_i - c_j) \equiv s((b_j - b_i) + m(d_j - d_i)) \pmod{r}, \\ s \equiv (a_i - a_j + m(c_i - c_j))((b_j - b_i) + m(d_j - d_i))^{-1} \pmod{r}. \end{gathered}$$

But, none of them can solve this equation because they do not know a_i, a_j, c_i, c_j.

The scheme is secure even if a user has access to the signatures of two distinct messages signed with the same keys because it implies solving IFP and DLP.

Finally, nobody can determine a forged multisignature for the message M without being detected by $\mathcal{T}$. In fact, a forger could know the public key, (P, Q), the message, M, its hash, m, the number of signers, t, and the values (α, r, β, n). From these data, he can choose an integer $\bar{g}$ and determine the element $\beta^{\bar{g}} = \alpha^{s\cdot\bar{g}} \in S_r$. Moreover, he can compute

$$\overline{F} \equiv P^t \cdot Q^{t\cdot m} \cdot (\beta^{\bar{g}})^{-1} \pmod{n}$$

and publish the pair $(\overline{F}, \bar{g})$ as a multisignature of the signer group G for the message M, that passes the verification equation (2).

Nevertheless, $\mathcal{T}$ can prove that this multisignature is a forgery. It is sufficient that it calculates

$$\widetilde{F} \equiv \prod_{i=1}^{t} A_i \cdot C_i^m \pmod{n},$$

and shows that $\widetilde{F}^{-1} \cdot \overline{F} \not\equiv 1 \pmod{n}$.

3 Conclusions

A new semi-short multisignature scheme based on three difficult problems from Number Theory, namely, integer factorization, discrete logarithms, and subgroup discrete logarithms has been proposed. A multisignature (F, g) is semi-short in the sense that $F \in \mathbb{Z}_n^*$ and $g \in \mathbb{Z}_r^*$, where the bitlength of n is much bigger than the the bitlength of r.

This scheme permits one to obtain a semi-short signature with a fixed bitlength, which is independent of the number of signers.

The multisignature scheme is efficient since the computations only require polynomial time, verifies the conditions of multisignature schemes, and moreover it is secure both against conspiracy attacks and against forgery.

Acknowledgment. This work has been partially supported by the "Fundación Memoria D. Samuel Solórzano Barruso" under the Project FS/7-2010.

References

1. Menezes, A., van Oorschot P., Vanstone, S.: Handbook of applied cryptography. CRC Press, Boca Raton, Florida (1997)
2. National Institute of Standards and Technology: Secure Hash Standard (SHS). Federal Information Processing Standard Publication 180-2 (2002)
3. Aboud, S.J.: Two efficient digital multisignature schemes. Int. J. Soft. Comput. 2, 113–117 (2007)
4. Boyd, C.: Some applications of multiple key ciphers. In: Günter, C.G. 8ed.) EUROCRYPT 1988. LNCS, vol. 330, pp. 445–467. Springer, Heidelberg (1988)
5. Itakura, K., Nakamura, K.: A public-key cryptosystem suitable for digital multisignatures. NEC Res. Development 71, 1–8 (1983)
6. Okamoto, T.: A digital multisignature scheme using bijective public-key cryptosystems. Commun. ACM Trans. Computer Systems 6, 432–441 (1988)
7. Aboud, S.J., Al-Fayoumi, M.A.: A new multisignature scheme using re-encryption technique. J. Applied Sci. 7, 1813–1817 (2007)
8. Harn, L., Kiesler, T.: New scheme for digital multisignature. Elect. Lett. 25, 1002–1003 (1989)
9. Kiesler, T., Harn, L.: RSA blocking and multisignature schemes with no bit expansion. Elect. Lett. 26, 1490–1491 (1990)
10. Park, S., Park, S., Kim, K., Won, D.: Two efficient RSA multisignature schemes. In: Han,Y., Quing, S. (eds.) ICICS 1997. LNCS, vol. 1334, pp. 217–222. Springer, Heidelberg (1997)
11. Pon, S.F., Lu, E.H., Lee, J.Y.: Dynamic reblocking RSA-based multisignatures scheme for computer and communication networks. IEEE Comm. Let. 6, 43–44 (2002)
12. Laih, C.S., Yen, S.M.: Multisignature for specified group of verifiers. J. Inform. Sci. Engrg. 12, 1, 143–152 (1996)
13. He, W.H.: Weakness in some multisignature schemes for specified group of verifiers. Inform. Proc. Lett. 83, 95–99 (2002)
14. Yen, S.M.: Cryptanalysis and repair of the multi-verifier signature with verifier specification. Computers & Security 15, 6, 537–544 (1996)
15. Zhang, Z., Xiao, G.: New multisignature scheme for specified group of verifiers. Appl. Math. Comput. 157, 425–431 (2004)
16. Lv, J., Wang, X., Kim, K.: Security of a multisignature scheme for specified group of verifiers. Appl. Math. Comput. 166, 58–63 (2005)
17. Yoon, E.J., Yoo, K.Y.: Cryptanalysis of Zhang-Xiao's multisignature scheme for specified group of verifiers. Appl. Math. Comput. 170, 226–229 (2005)
18. Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: Proc. 13th ACM conference on Computer and Communications Security (CCS'06), pp. 390–399. ACM Press, New York (2006)
19. Pedersen, T.P., Pfitzmann, B.: Fail-stop signatures, SIAM J. Comput. 26, 291–330 (1997)
20. Maurer, U.: Some number-theoretic conjectures and their relation to the generation of cryptographic primes. In: Proc. Cryptography and Coding'92, pp. 173–191. Oxford University Press, New York (1992)
21. Susilo, W.: Short fail-stop signature scheme based on factorization and discrete logarithm assumptions. Theor. Comput. Sci. 410, 736–744 (2009)