Undetachable threshold signatures

Niklas Borselius Chris J. Mitchell
Aaron Wilson
Mobile VCE Research Group,
Information Security Group,
Royal Holloway, University of London
Egham, Surrey TW20 0EX, UK

July 15, 2001

Abstract

A major problem of mobile agents is their inability to authenticate transactions in a hostile environment. Users will not wish to equip agents with their private signature keys when the agents may execute on untrusted platforms. Undetachable signatures were introduced to solve this problem by allowing users to equip agents with the means to sign signatures for tightly constrained transactions, using information especially derived from the user private signature key. However, the problem remains that a platform can force an agent to commit to a sub-optimal transaction. In parallel with the work on undetachable signatures, much work has been performed on threshold signature schemes, which allow signing power to be distributed across multiple agents, thereby reducing the trust in a single entity. We combine these notions and introduce the concept of an undetachable threshold signature scheme, which enables constrained signing power to be distributed across multiple agents, thus reducing the necessary trust in single agent platforms. We also provide an RSA-based example of such a scheme based on a combination of Shoup's threshold signature scheme, [7] and Kotzanikolaou et al's undetachable signature scheme, [3].

1 Introduction

A digital signature is the electronic counterpart to a written signature. Thus one way to commit to an electronic transaction is by the use of a digital signature. Recently, the use of mobile agents to commit to transactions for a user has become a topic of interest. Mobile agents, however, face the problem of having to execute in a hostile environment where the host executing the agent has access to all the data that an agent has stored (for instance the private signature key). Consequently, the problem of allowing an agent to sign a transaction on behalf of a user is one of interest.

Undetachable signatures were first proposed by Sander and Tschudin [5] to solve this problem, and are based on the idea of computing with encrypted functions. The host executes a function $s \circ f$, where f is an encrypting function, without having access to the user's private signature function s. The security of the method lies in the encrypting function f. Whilst Sander and Tschudin were unable to propose a satisfactory scheme, more recently Kotzanikolaou, Burmester and Chrissikopoulos [3] have presented an RSA-based scheme which appears to be secure.

The idea of an undetachable signature is as follows. Suppose a user wishes to purchase a product from an electronic shop. The agent can commit to the transaction only if the agent can use the signature function s of the user. However as the server where the agent executes may be hostile, the signature is protected by a function f to obtain $g = s \circ f$. The user then gives the agent the pair (f, g) of functions as part of its code. The server then executes the pair (f, g) on an input x (where x encodes transaction details) to obtain the undetachable signature pair

$$f(x) = m \text{ and } g(x) = s(m).$$

The pair of functions allows the agent to create signatures for the user whilst executing on the server without revealing s to the server. The parameters of the function f are such that the output of f includes the user's constraints. Thus m links the constraints of the customer to the bid of the server. This is then certified by the signature on this message. The main point is that the server cannot sign arbitrary messages, because the function f is linked to the user's constraints.

However, one problem with this approach is that the agent is still given the power to sign any transaction it likes, subject to the requirement that the transaction must be consistent with the constraints used to construct f. Thus, for example, whilst the constraints may limit the nature and/or value of a transaction, a malicious host may force an agent to commit to a transaction much less favourable than could be achieved.

Thus, to protect further against malicious hosts, a user may wish to use more than one agent and have the agents agree on a bid before committing to it. Hence, a user may send out n agents with the criteria that k of them must agree before committing to a purchase. The obvious solution to such a requirement is to employ a *threshold signature scheme*, meaning that agents can all sign the bid they think ‘best’ given the user’s requirements, and then, on receipt of a sufficient number of these bids, the user’s signature can be reconstructed.

However, such a scheme does not possess the means to constrain the power given to a quorum of agents. This motivates the introduction of the concept of an *undetachable threshold signature* which both distributes signature authority across multiple agents and simultaneously constrains the signatures that may be constructed.

The rest of the paper is as follows. In Section 2 we outline the undetachable signature scheme of [3], and in Section 3 we briefly review threshold signatures and give a method of Shoup [7] to construct such a scheme. Finally, in Section 4 we define the concept of an undetachable threshold signature, and show how an example of such a scheme may be obtained by combining the schemes of [3] and [7].

2 RSA Undetachable signatures

We briefly present the RSA undetachable signature scheme given in [3]. The user sets up an RSA signature pair in the usual manner, that is the user selects an RSA modulus n which is the product of two primes p and q, and a number e such that $1 \leq e \leq \phi(n) = (p-1)(q-1)$ and $\gcd(e, \phi(n)) = 1$. Let d be such that $1 \leq d \leq \phi(n)$ and $ed = 1 \bmod \phi(n)$. The user then publishes the verification key (n, e) and keeps d as the private signing key.

Let I be an identifier for the user and R the encoded requirements of the user for a purchase (we assume that R is encoded in a manner which is understood by all parties). Let h be an appropriate hash-function (*i.e.* one giving a value in $\mathbb{Z}_n$). The user then forms $H = h(I, R)$.

The user then gives an agent the user identifier, the requirements, and the pair (H,G) as its undetachable signature, where $G = H^d \bmod n$. To sign a bid B (which we assume is in the same format as R), the executing host calculates $x = h(B)$. The undetachable signature is then the pair (H^x, G^x). We note that,

$$G^x = (H^d)^x = H^{dx} = H^{xd} = (H^x)^d$$

so that the server has signed the value H^x with the user’s private key.

We briefly note that this scheme appears secure, and a proof of this fact is given in [3]. To forge a signature on a different set of requirements R' a malicious host would need to forge $H' = h(I, R')$, $G' = (H')^d$ and $(G')^x$. Clearly the only work needed here is to forge G', and this would require knowledge of a user's private key. Having said this, there is nothing in this scheme to prevent a host from signing more than one bid, or presenting a bid that just meets the requirements of the user (as opposed to a possibly better standard offer).

3 Threshold Signatures

The idea of a threshold scheme is to take a secret, and divide it into pieces called shares which are distributed among a group of entities. Then any subset of these entities of a given size can reconstruct this secret, but a smaller group can learn no information about the secret. An example of such a scheme is given in [6].

Threshold cryptography was first proposed by Desmedt [2]. One important type of threshold cryptosystem is known as a *threshold signature*. In such a scheme, any set of k parties from a total of l parties can sign any document, and any coalition of less than k parties cannot sign a document. Such schemes tend to rely on a combiner which is not necessarily trusted. Several schemes have been proposed based on both El Gamal and RSA cryptography (see, for example, [7] for a short survey). Recently Shoup [7] proposed an RSA scheme which is as efficient as possible; the scheme uses only one level of secret sharing, each server sends a single part signature to a combiner, and must do work that is equivalent, up to a constant factor, to computing a single RSA signature.

Although in some sense not perfect as a threshold signature scheme (as it relies on a trusted party to form the shares) this scheme is ideal in our setting, where the user dispatching the agent will always (one would hope) trust themselves. (Note that an alternative scheme without a trusted dealer is given in [1]. This scheme also improves on [7] by not relying on an RSA modulus made up of 'safe primes'). An example of an El Gamal based scheme is given in [4].

We next briefly outline the threshold signature scheme of [7].

The user (dealer) forms the following:

- An RSA modulus $n = pq$ where $p = 2p' + 1$ and $q = 2q' + 1$ are safe primes, *i.e.* p', q' are prime.

- A public exponent e where e is prime and a private key d, where $de \equiv 1 \pmod{p'q'}$.
- A polynomial $f(x) = \sum_{i=0}^{k-1} a_i x^i$ where $a_0 = d$ and $a_i \in \{0, \ldots, p'q' - 1\}$ (selected at random) for $1 \leq i \leq k$.
- $L(n)$, the bit length of n, and L_1, a secondary security parameter — Shoup [7] suggests $L_1 = 128$.
- The l signature key shares of the scheme s_i, where each s_i is selected at random from the set $\{s | 0 \leq s \leq 2^{L(n)+L_1}, s \equiv f(i) \bmod (p'q')\}$.
- The verification keys $\mathrm{VK} = v$ and $\mathrm{VK}_i = v^{s_i}$ where $v \in Q_n$, the subgroup of squares of $\mathbb{Z}_n^*$.
- A global hash function h mapping into $\mathbb{Z}_n^*$.
- A second hash function g whose output is an L_1-bit integer.

In this scheme a shareholder signs a message m in the following manner. Firstly the shareholder calculates the hash of the message, i.e. $x = h(m)$. The signature share of a shareholder i then consists of

$$x_i = x^{2\Delta s_i}$$

and a 'proof of correctness' (note that $\Delta = l!$). The proof of correctness is basically just a proof that the discrete logarithm of x_i^2 to the base $x^{4\Delta}$ is the same as the discrete logarithm of v_i to the base v. Let $L(n)$ be the bit length of n. The shareholder then chooses a random number $r \in \{0, \ldots, 2^{L(n)+3L_1} - 1\}$ and computes

$$v' = v^r,\ x' = x^{4\Delta} r,\ c = g(v, x^{4\Delta}, v_i, v', x'),\ z = s_i c + r.$$

The proof of correctness is then (z, c) which can be verified by calculating

$$c = g(v, x^{4\Delta}, v_i, x_i^2, v^z v_i^{-c}, x^{4\Delta z} x_i^{-2c}).$$

To combine the shares the combiner acts as follows. Assume we have valid shares from a set $S = \{i_1, i_2, \ldots, i_k\}$ of shareholders. The combiner computes

$$\lambda_{0,j}^S = \Delta \prod_{i \in S \setminus \{j\}} \frac{i}{(i-j)}.$$

These values are derived from the standard Lagrange interpolation formula. These values are integers and it is clear that they are easy to compute. We also have, from the Lagrange interpolation formula that,

$$\Delta \cdot f(0) = \sum_{j \in S} \lambda_{0,j}^S f(j) \bmod (p'q').$$

In other words we have,

$$d \cdot \Delta = \sum_{j \in S} \lambda_{0,j}^{S} s_j$$

The combiner then computes,

$$\begin{aligned} w &= x_{i_1}^{2\lambda_{0,i_1}^{S}} \cdots x_{i_k}^{2\lambda_{0,i_k}^{S}} \\ &= x^{4\Delta^2 \sum_{j \in S}(s_j \lambda_{0,j}^{S})} \\ &= x^{4\Delta^5 d}. \end{aligned}$$

To check this signature we note that $w^e = x^{4\Delta^5}$ where $\gcd(e, 4\Delta^5) = 1$. As e is coprime to $4\Delta^5$ we can find a, b such that $a(4\Delta^5) + be = 1$ so that we finally have the signature

$$y^e = (w^a x^b)^e = x.$$

4 Undetachable Threshold Signatures

We now introduce the notion of an undetachable threshold signature. Suppose a user has a private signature key s and a public verification key v. Suppose also that the user has a 'constraint string' R, which will define what types of signature can be created. Then an undetachable threshold signature scheme will enable the user to provide n entities with 'shares' of the private signature key (where the shares will be a function of R), where the following properties must be satisfied:

- each entity can use their share to sign a message M of their choice to obtain a 'signature share';
- the 'correctness' of a signature share can be verified independently of any other signature shares;
- any entity, when equipped with k different signature shares for the same message M, can construct a signature on the message M which will be verifiable by any party with a trusted copy of the public key of the user, and which will also enable the string R to be verified;
- knowledge of less than k different signature shares for the same message M cannot be used to construct a valid signature on the message M;

- knowledge of any number of different signature shares for messages other than M will not enable the construction of a valid signature on message M;
- knowledge of any number of different signature shares for constraints strings other than R will not enable the construction of a valid signature with associated constraint string R.

As discussed above, the motivation for introducing this concept is that the use of a threshold signature scheme or a detachable signature scheme on its own would not protect against all possible attacks in a mobile agent scenario. We now describe an example of such a scheme. For brevity, we only give the necessary changes to the threshold scheme in section 3 to form the undetachable threshold signature scheme.

Recall that the secret share for shareholder i consists of a number s_i. Let h be an appropriate hash function. The signature share of this shareholder for a message m is then

$$x_i = x^{2 \cdot \Delta \cdot s_i}.$$

where l is the total number of shares, $\Delta = l!$ and $x = h(m)$ is a hash of the message.

As in Section 2 let I be the identifier of a user and let R be the user requirements. Let $H = h(I, R)$ be a hash of the requirements. We replace the share s_i with a pair $(H, t_i = H^{2 \cdot \Delta \cdot s_i})$. To sign a bid B the shareholder calculates $C = h(B)$ and

$$t_i^C = (H^{2 \cdot \Delta \cdot s_i})^C = H^{2 \cdot \Delta \cdot s_i C} = (H^C)^{2 \cdot \Delta \cdot s_i}.$$

Thus, when all the shares are combined the combiner will have a signed copy of H^C, thus achieving a signed undetachable signature.

We observe that a proof of security is given for the scheme in Section 3 provided that k is one greater than the number of corrupt servers (in the case where k exceeds the number of corrupt servers by a greater number a slightly adapted scheme is used). With this information to hand we note that this scheme is secure as long as the undetachable scheme given in [3] is secure, and that this scheme appears to be sound.

Acknowledgements

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

References

[1] Ivan Damgård and Maciej Koprowski. Practical threshold RSA signatures without a trusted dealer. In *Proceedings of EuroCrypt 2001*, to appear.

[2] Y. Desmedt. Society and group oriented cryptography. In C. Pomerance, editor, *Advances in Cryptology – Crypto '87 proceedings*, number 293 in LNCS, pages 120–127. Springer-Verlag, 1988.

[3] Panayiotis Kotzanikolaou, Mike Burmester, and Vassilios Chrissikopoulos. Secure transactions with mobile agents in hostile environments. In E. Dawson, A. Clark, and C. Boyd, editors, *Information Security and Privacy, Proceedings of the 5th Australasian Conference ACISP 2000*, number 1841 in LNCS, pages 289–297. Springer-Verlag, 2000.

[4] Susan K. Langford. Threshold DSS signatures without a trusted party. In D. Coppersmith, editor, *Advances in Cryptology – Crypto '95 proceedings*, number 963 in LNCS, pages 397–409. Springer-Verlag, 1995.

[5] Tomas Sander and Christian Tschudin. Protecting mobile agents against malicious hosts. In Giovanni Vigna, editor, *Mobile Agents and Security*, number 1419 in LNCS. Springer-Verlag, 1998. Available from http://www.icsi.berkley.edu/ sander/publications/MA-protect.ps.

[6] A. Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, 1979.

[7] Victor Shoup. Practical threshold signatures. In Bart Preneel, editor, *Proceedings of EuroCrypt 2000*, number 1807 in LNCS, pages 207 –220. Springer-Verlag, 2000.