On the value of threshold signatures*

Niklas Borselius, Chris J. Mitchell and Aaron Wilson
Mobile VCE Research Group, Information Security Group
Royal Holloway, University of London
Egham, Surrey TW20 0EX, UK
niklas.borselius@rhul.ac.uk, aaron@gmx.co.uk, c.mitchell@rhul.ac.uk

August 6, 2002

Abstract

Threshold signature schemes are examples of threshold cryptosystems, as introduced by Desmedt, [4]. The purpose of this paper is to present a rather simple alternative to threshold signatures which raises questions about the value of such schemes, at least when applied to the mobile agent scenario.

Keywords: mobile agent, digital signature, threshold signature, security

1 Introduction

Threshold signature schemes enable a group of n entities to be given 'shares' of a private signature key in such a way that, for some parameter k ($1 \leq k \leq n$), any subset of k entities can collectively create a valid signature on a message, whereas any collection of $k-1$ or fewer entities cannot. Schemes of this type have been discussed widely in the literature, and a number of systems have been proposed; see for example [5] for a brief survey.

One particularly attractive scheme has been recently proposed by Shoup, [5]. This scheme is based on RSA, and the composite signature can be verified in exactly the same way as a 'regular' RSA signature. In the discussion below we use this scheme as a 'benchmark' against which alternatives can be compared.

*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.

2 Mobile agents and threshold signatures

Much research in recent years has been devoted to the problem of providing security for mobile agents. The problem is usually divided into two parts:

- protecting the platform on which agents run against malicious and/or unauthorised agents,
- protecting the agents against malicious platforms.

The focus of this note is the second problem, which is usually regarded as the most difficult one to solve.

Of course, there are limits to the protection that can be offered to an agent. An agent platform can potentially modify the agent code, and/or interfere with the data stored by an agent. Hence efforts to protect agents reduce to either finding ways to enhance the level of trust that can be placed in results produced by an agent, or limiting the powers given to an agent. This paper focuses on the former approach — in particular it considers the issue of dividing a task amongst multiple agents to increase the level of trust in the collective results of the agents.

In particular we are concerned with giving all subsets of agents of size k or more the power to sign on a user's behalf, without giving smaller groups of agents such a capability. We suppose that the agents are set up to agree to a transaction, the details of which the agents are to determine. This might typically occur by giving the agents the power to visit a number of merchants, and find out which merchant is offering a particular good or service at the lowest price. Some of the agents may be modified by a malicious host, but we assume that this occurs to at most $k-1$ of them.

As discussed in more detail in [1], this seems to be a natural application for the notion of threshold signatures. The user launching the agents acts as the 'dealer' in the threshold signature scheme, and creates the shares of the signature key. Each agent is equipped with a share, and when a transaction is to be signed creates a signature share. A third party (e.g. the merchant making the transaction) receives the signature shares and uses them to construct a valid signature. This distribution of trust across a multiplicity of agents reduces the threat from small numbers of malicious hosts, who might be capable of manipulating agent computations. Finally note that this problem also relates to the well-known concept of a *multisignature* which, for our purposes at least, can be regarded as a special type of threshold signature for the case $k = n$.

3 An alternative based on conventional signatures

We now consider an alternative solution (in the spirit of [2]) to this trust distribution problem. Unlike the use of threshold signatures, this solution is wholly based on widely used cryptographic primitives, namely digital signatures, and hence may be more likely to succeed in practice. It is also quite general in its specification, allowing the use of any digital signature scheme.

Note that it is implicit to the solution described immediately below, and also to the solutions using threshold signatures, that the agents are transferred by the user U to the hosts on which they are to execute by some secure means. This secure transfer should enable the receiving host to check its integrity and origin, and also should protect the confidentiality of all the sensitive parts of the agent (most crucially including any embedded secret or private keys).

3.1 Preparing the agents

Before sending the agents, (which we label $A_1, A_2, \ldots, A_n$) user U performs the following steps. Note that we assume that U has a signature key pair of its own, (S_U, P_U) say, and a certificate Cert_U for its own public key, P_U, signed by a Certification Authority (CA).

1. U generates a signature key pair (S_i, P_i) specifically for use by agent A_i.
2. U creates a public key certificate Cert_i for agent A_i's public key (P_i), signed using U's own signature key (S_U). This certificate also contains policy information which indicates precisely the purpose of the certificate, and also the parameter k, indicating the 'threshold value' (as above). It is also likely that this certificate will have a very short lifetime, i.e. it would have an expiry date very close to the time of issue.
3. U now equips agent A_i with the private signature key S_i, and copies of the two certificates Cert_i and Cert_U.
4. A_i is now securely transferred to one or more agent platforms.

3.2 Executing an agent

Now suppose that some subset of k agents decide that they wish to collectively sign a message (e.g. to commit to a transaction with a merchant) on behalf of user U. Each agent A_i signs the message using the agent's private key S_i, and the signature is then transferred with the two certificates Cert_i and Cert_U to the entity requiring the signature, e.g. a merchant. The recipient of the agent signatures, M say, then performs the following steps for each received agent signature.

1. The user's certificate Cert_U is verified by M using a trusted copy of the CA's public key. If M does not have this CA's public key, then it will need to be derived by some means, e.g. using a certificate chain. Next, M then checks that it is prepared to accept a signature from U, and also checks that the name in the certificate is consistent with the user name received from the agent.

   Note that this step will only need to be performed once for the k agent signatures.
2. The agent's certificate Cert_i is verified by M using the copy of U's public key obtained in the first step.
3. The agent finally checks the agent's signature using the copy of A_i's public key obtained from Cert_i.

The merchant waits until k valid agent signatures have been received (recall that k was encoded in each agent certificate). The collection of k agent signatures is then deemed to be equivalent to the signature of the user, and the merchant proceeds with the transaction. The collection of k agent signatures (with the accompanying certificates) are retained as evidence of the transaction.

3.3 Remarks on implementation

Before attempting to compare this new scheme with the use of threshold signatures we make some remarks about the implementation of the scheme.

- Given that the agent key pairs have only a short lifetime, it may be possible to use relatively short keys. That is, if the signature scheme is RSA based, a short modulus could be used, say of 512 bits, in the knowledge that factoring the modulus and hence breaking the key would be infeasible during the key's lifetime. This would make key generation faster and would reduce the amount of key information to be transferred. It would also mean that creating and verifying agent signatures could be made significantly more efficient.

- If a 'weak' key pair was used for an agent key, or, more generally, if the certificate for an agent key pair has a very short period of validity, problems might arise if the agent's signature is required to have long term validity, e.g. to provide a non-repudiation service in the event of a dispute. The 'standard' way of resolving this problem is to use a timestamping service to sign a concatenation of the signature and a timestamp, providing evidence that the signature was generated during the key's period of validity. An alternative to using a trusted timestamping service would be to simply require the agent host to add a timestamp and its signature to any signatures output by the agent. Not only would this provide evidence about when the agent signed the message, but it would also enable the merchant receiving the signature to verify on which host the agent was running. This would appear to be a valuable service in its own right.

- The scheme as described does not restrict the nature (e.g. value and/or type) of messages which the agents can collectively sign. However, a restriction could easily be imposed by including the scope of the agent keys in the respective agent public key certificates.

- This scheme is in some respects analogous to the widely discussed notion of delegation using special delegation keys — see, for example, [3] for an introduction to delegation issues and [6] for one approach to the use of delegation keys.

3.4 A brief comparison

We now attempt to briefly compare the efficiency of the above scheme with the efficiency achievable using the Shoup threshold signature scheme, [5]. For the purposes of the comparison we suppose that signatures for the scheme in this paper are computed using RSA, and we compare this with the Shoup RSA-based threshold signature scheme, [5]. To compare the efficiencies of the two schemes we compare separately the work to be performed by the user U, each agent A_i, and the recipient of the agent signatures, M.

- *User* U. For the scheme above, the user will be required to generate one key pair for each agent A_i, and certify the public keys, i.e. compute n signatures and generate n key pairs. For the threshold signature scheme of [5], the user is only required to generate at most one key pair (for the secret key that is shared by the n agents). However, each agent will need to be equipped with a public key certificate for its share, so that the signature share can be

verified by the merchant (or whoever combines the signature shares). Hence the user will be required to generate one key pair and compute n signatures. Note that, whilst key generation will typically take much longer than computing a signature, key pairs could not only be made quite small (as discussed above), but could be generated in advance. Hence, the new scheme, whilst requiring more computation overall, actually requires comparable amounts of computation at the time of agent creation.

- *Agent A_i.* For the new scheme, the agent is required to compute one signature. For the Shoup scheme, each agent is required to perform one exponentiation, equivalent to one signature.

- *Recipient of agent signatures M.* For the new scheme the recipient of the signed message will be required to verify $k + 1$ certificates (the user certificate and the k agent certificates) and k signatures, i.e. a total of $2k + 1$ signature verifications. For the Shoup scheme it is also necessary to verify the user's certificate, as well the k agent certificates and the k signature shares. Hence the two schemes have roughly comparable computational efficiencies. Of course, the storage efficiency for the scheme described above will be rather less than for the Shoup scheme, unless it is necessary to retain the signature shares for auditing purposes.

In summary it would appear that, in efficiency terms, the scheme of this paper is really quite comparable with the Shoup threshold signature scheme, with two exceptions. Initialising the scheme requires a number of key generations, which, however, could be done 'off-line'. The other difference is in the size of storage required for signatures, which is significantly larger for the new scheme.

However, this advantage disappears if the signature shares must be kept for auditing purposes. Indeed, this is not such an unlikely scenario, since, in the event of a dispute, it will be valuable to learn which agents took part in the signing process. Thus the implementation efficiency differences, which could be rather minor in practice, could easily be outweighed by the advantages inherent in using established cryptographic primitives.

Finally observe that one other difference between the approach proposed here and the use of threshold signatures relates to the issues of anonymity and accountability. In most threshold signature schemes, the verifier of the signature cannot determine which of the k shareholders created the signature, whereas with the above approach there is no anonymity for agents. In some circumstances the anonymity property may be desirable, but in many agent applications the reverse is likely to be true. That is, the originator of the agents will in many cases wish to have the means to determine which agents performed actions on its behalf.

4 Concluding remarks

A pragmatic alternative to 'threshold signatures' problem has been proposed. This alternative has potential practical advantages by comparison with the use of threshold signatures, and appears to offer a very similar set of security guarantees. Although the analysis was performed within the context of mobile agents, it is possible that the scheme described here is competitive with the Shoup threshold signature scheme in other environments.

References

[1] N. Borselius, C.J. Mitchell, and A.T. Wilson. On mobile agent based transactions in moderately hostile environments. In B. De Decker, F. Piessens, J. Smits, and E. Van Herreweghen, editors, *Advances in Network and Distributed Systems Security, Proceedings of IFIP TC11 WG11.4 First Annual Working Conference on Network Security, KU Leuven, Belgium, November 2001*, pages 173–186. Kluwer Academic Publishers, Boston, 2001.

[2] N. Borselius, C.J. Mitchell, and A.T. Wilson. A pragmatic alternative to undetachable signatures. *ACM SIGOPS Operating Systems Review*, 36(2):6–11, April 2002.

[3] B. Crispo. Delegation of responsibility (position paper). In B. Christianson, B. Crispo, W.S. Harbison, and M. Roe, editors, *Security protocols: 6th International Workshop, Cambridge, UK*, number 1550 in Lecture Notes in Computer Science, pages 118–124. Springer-Verlag, Berlin, 1998.

[4] Y. Desmedt. Society and group oriented cryptography: A new concept. In C. Pomerance, editor, *Advances in Cryptology — CRYPTO '87*, number 293 in Lecture Notes in Computer Science, pages 120–127. Springer-Verlag, Berlin, 1988.

[5] V. Shoup. Practical threshold signatures. In B. Preneel, editor, *Advances in Cryptology — EUROCRYPT 2000*, number 1807 in Lecture Notes in Computer Science, pages 207–220. Springer-Verlag, Berlin, 2000.

[6] V. Varadharajan, P. Allen, and S. Black. An analysis of the proxy problem in distributed systems. In *Proceedings: 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 255–275. IEEE Computer Society Press, Los Alamitos, California, May 1991.