A pragmatic alternative to undetachable 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

6th January 2002

Abstract

A 'pragmatic' alternative to undetachable signatures is proposed. Undetachable signatures were introduced by Sander and Tschudin, [4], as a means of giving a mobile agent the means to sign a message on behalf of a user, without endangering the user's private key. The alternative discussed in this paper involves the use of conventional signatures and public key certificates.

Keywords: Cryptography, distributed systems

1 Introduction

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 latter approach — in particular it considers the issue of giving an agent the power to sign on a

[*]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.com`, 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.

user's behalf, without running the risk of exposing the user's private signature key to the platform on which the agent is executed.

The notion of an undetachable signature was introduced by Sander and Tschudin in [4]. This provides a means of providing an agent with a signature key 'derived from' the user's own signature key. Signatures produced using the derived key can be verified by anyone with access to a reliable copy of the user's public verification key (e.g. as obtained from a public key certificate for the user's key). Moreover, the user can embed within the derived key a statement as to the circumstances in which the derived key may be used, e.g. to state what types of goods might be purchased and/or to state a price limit.

Whilst the scheme proposed in [4] has proven insecure, an alternative RSA-based scheme proposed by Kotzanikolaou, Burmester and Chrissikopoulos, [3], appears to be sound. However, it is a new scheme, and should perhaps be used with care. In the remainder of this paper we describe an alternative approach which uses only well-established cryptographic techniques.

2 Solving the problem the conventional way

We now consider an alternative solution to the problem which undetachable signatures have been introduced to solve. This solution is wholly based on conventional cryptographic primitives, 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.

2.1 Preliminaries

Suppose user U wishes to create a mobile agent A that may run on one or more agent platforms not completely trusted by U. Suppose also that user U wishes to give A the power to sign statements on behalf of U, as long as the statement conforms to rules specified in a string R (where R is in a form agreed by all parties to the transaction). It is possible that R may be completely explicit about the rules governing the signing process by the agent, or R may simply contain one or more pointers to generally agreed policy statements, perhaps with additional parameters.

Note that it is implicit to the solution described immediately below, and also to the solutions using undetachable signatures, that the agent is transferred by U to the platform on which it is to execute by some secure means. This secure transfer should enable the receiving agent platform 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).

2.2 Preparing the agent

Before sending the agent A, 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_A, P_A) specifically for use by the agent.

2. U creates a public key certificate Cert_A for the agent's public key (P_A), signed using U's own signature key (S_U). This certificate also contains a copy of the string R, which states in what circumstances A's key may be used, and which makes it clear that P_A is an agent key. It is also expected that this certificate would have a very short lifetime, i.e. it would have an expiry date very close to the time of issue.

3. U now equips the agent A with the private signature key S_A, and copies of the two certificates Cert_A and Cert_U.

4. A is now securely transferred to one or more agent platforms.

2.3 Executing the agent

When A executes, it may be necessary for A to sign a message of some kind (e.g. a commitment to a transaction) on behalf of user U. Such a transaction should be signed using the agent's private key S_A, and the signature should then be transferred with the two certificates Cert_A and Cert_U to the entity requiring the commitment, e.g. a merchant. The recipient of the commitment, M say, then performs the following steps.

1. The user's certificate Cert_U is verified by M using a trusted copy of the CA's public key. Note that 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.

2. The agent checks that it is prepared to accept a commitment from U, and also checks that the name in the certificate is consistent with the user name received from the agent.

3. The agent's certificate Cert_A is verified by M using the copy of U's public key obtained in the first step.

4. The agent checks that the string R contained in Cert_A is consistent with the transaction that is taking place.

5. The agent finally checks the agent's signature using the copy of A's public key obtained from Cert_A.

It should be clear that, by the simple step of including R in the certificate for A's key pair, the power given to A by U can be limited to that specified by U. This has been achieved without any need for a new cryptosystem.

2.4 Remarks on implementation

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

- Given that the agent key pair has only a short lifetime, it may be possible to use a relatively short key. 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 the agent key, or, more generally, if the certificate for the 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 platform to add a timestamp and its signature to any signed commitments output by the agent. Not only would this provide evidence about when the agent signed the message, but it would also enable the recipient of the signed message to verify on which platform the agent was running. This would appear to be a valuable service in its own right, which would apply equally to the case where an undetachable signature scheme is employed.

2.5 A brief comparison

We now attempt to briefly compare the efficiency of the above scheme with the efficiency achievable using an undetachable signature scheme. For the purposes of the comparison we suppose that signatures for the scheme in this paper are computed using RSA and a hash-function, and we compare this with the RSA-based undetachable signature scheme of Kotzanikolaou et al., [3]. To compare the efficiencies of the two schemes we compare separately the work to be performed by the user U, the agent A, and the recipient of the commitment, M.

- *User U.* For the scheme above, the user will be required to generate a key pair and certify the public key, i.e. compute one signature and generate one key pair. For the undetachable signature scheme of [3], the user is required to perform two exponentiations, equivalent in complexity to performing two 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 less computation at the time of agent creation.

- *Agent A.* For the new scheme, the agent is required to compute one signature. For the undetachable signature scheme, the agent is required to perform two exponentiations, equivalent to two signatures. Moreover, for the undetachable signature scheme these will be 'full size' signatures, whereas for the new scheme the key lengths may be reduced (as above).

- *Recipient of commitment M.* For the new scheme the recipient of the signed message will be required to verify two certificates and a signature, i.e. a total of three signature verifications. For the undetachable signatures scheme it is also necessary to verify the user's certificate, as well as performing two exponentiations. Hence the two schemes have roughly comparable efficiencies.

It would appear that the scheme of this paper has potential efficiency advantages over the undetachable signature scheme, quite apart from the advantages inherent in using established cryptographic primitives.

2.6 Relationship to secure delegation schemes

Note that the problem which the proposed solution is designed to address appears to be closely related to the problem of secure delegation in distributed systems. Delegation refers to the situation where one entity wishes a separate entity to perform a task on its behalf. Security problems arise when the delegated entity does not have the access rights to perform the task, and hence must be temporarily given these rights in order to perform the requested actions. The issue then becomes one of giving these rights in such a way that they cannot be abused. See, for example, [1] for a general introduction to delegation issues.

An analogous approach to the one described here has been proposed by several authors as a solution to secure delegation — see, for example, [5]. However, instead of the use of a public key certificate, special 'delegation tokens' have been proposed. Note also that, as described in [2], issues can arise with any such solution since the originating user will have a copy of the private key generated for agent use. The user may use this key to masquerade as the agent, and then deny the transaction, blaming the platform on which the agent has run. The proposed use of countersignatures by the agent platform, as described in Section 2.4 above, significantly reduces the seriousness of this threat.

3 Concluding remarks

A pragmatic solution to a mobile agent security problem has been proposed. This solution has potential practical advantages by comparison with the use of undetachable signatures, and appears to offer a very similar set of security guarantees. When combined with the use of signatures by the agent platform, this solution has the potential to solve certain problems relating to transaction repudiation.

References

[1] 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.

[2] B. Crispo and B. Christianson. A note about the semantics of delegation. Preprint, 1999.

[3] P. Kotzanikolaou, M. Burmester, and V. 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 Lecture Notes in Computer Science, pages 289–297. Springer-Verlag, Berlin, 2000.

[4] T. Sander and C.F. Tschudin. Protecting mobile agents against malicious hosts. In G. Vigna, editor, *Mobile agent security*, number 1419 in Lecture Notes in Computer Science, pages 44–60. Springer-Verlag, Berlin, 1998.

[5] 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.