Certificates for Mobile Code Security

Hock Kim Tan
Department of Electronics and Computer Science
University of Southampton
Southampton SO17 1BJ, UK
hkvt99r@ecs.soton.ac.uk

Luc Moreau
Department of Electronics and Computer Science
University of Southampton
Southampton SO17 1BJ, UK
L.Moreau@ecs.soton.ac.uk

Keywords
Mobile agent security, mobile agent certificates, mobile agent security framework

ABSTRACT
The problem of protecting mobile code from malicious hosts is an important security issue, for which many solutions have been proposed. We describe a method to adapt an existing technique, execution tracing, to enhance its flexibility in deployment for a large scale mobile agent system. This is achieved through the introduction of a trusted third party, the verification server, which undertakes the verification of execution traces on behalf of the platform launching the agent. The server constructs a certificate that testifies to the capability of a particular host platform to undertake the correct execution of a mobile agent. In this sense, the server assumes a role analogous of a Certificate Authority (CA) in a PKI. We briefly discuss the issues associated with such a framework.

1. INTRODUCTION
Mobile agents have become popular in recent years, being viewed as a new paradigm for distributed computation that offers a more flexible alternative to traditional client-server computing environments. Several advantages have been identified [15] for utilizing mobile agents in a distributed environment. There has also been several application areas proposed for the wide scale deployment of mobile agents, the most notable being electronic commerce [9], [6]. In such an area, security becomes a paramount issue as financial contracts may be negotiated by the mobile agent without direct user intervention.

A conceptual discussion of the security concerns of mobile code is provided in [4]. In general, the two main areas of consideration are host security and code security. Host security involves protecting the host platform from malicious agents that may attempt to gain unauthorized access to local resources or initiate malicious actions on the platform. Code security is the exact reverse and aims to protect the mobile agent from a malicious host platform that may attempt to subvert its correct operation or manipulate sensitive data contained within.

Host security is a reasonably well researched issue, and many viable mechanisms have been developed to address it. Examples of such mechanisms include sandbox security (used in Java to provide access control) [8], safe-typed languages [20] and proof-carrying code [16]. Code security is however more problematic, since this aspect has only come into prominence recently as a security problem unique to mobile code. Some of the more well known mechanisms used to overcome this problem include code obfuscation [10], encrypted functions [17], execution tracing [19] and tamper resistant hardware [21], [23]. It is likely this area will be crucial in determining the future viability of mobile agent application in e-commerce scenarios. An overview of the current techniques available to address both host and code security is provided in [12] and [22]; the reader is referred to them for a summary of these techniques as well as a discussion of their corresponding advantages and drawbacks.

The majority of these techniques were designed as standalone solutions to be deployed independently of existing security infrastructures for large scale environments, such as a Public Key Infrastructure (PKI) [1]. Integration of a technique within a PKI would greatly ease its incorporation into a security framework for a real-world mobile agent application scenario (such as e-commerce). The main reason underlying the lack of attempts at such an integration is because a PKI is primarily designed to address the issue of distributed authentication. This may be useful for host security by verifying the identity of an agent or its deployer, but it is not immediately obvious how authentication can aid in the protection of mobile code.

In this paper, we make an argument for the inclusion of trust as an explicit component of a security framework for mobile code. In the context of our work, we adopt the popular definition of trust [3] as a necessary counterpart to the delegation of tasks and/or responsibilities in an agent system. A system involving the use of trust (such as a PKI) usually includes a trust model that formally describes how trust is manipulated and propagated in that system. Incorporating trust in a code security technique permits the development of such a trust model, which will be useful for

SAC 2002, Madrid, Spain
@2002 ACM 1-58113-445-2/02/03 ... $5.00

indepth analysis on the integration process with a PKI. We present execution tracing as a code security technique suitable for establishing trust levels (i.e. quantitative measure of trust) in such a framework. We then show how this technique can be extended so that it incorporates a trusted third party that utilizes certificates that are distributed and managed within the context of a PKI. Our main contributions are thus:

- Arguing for trust as a necessary component in a framework to address mobile code security and the use of the execution tracing technique to establish levels of trust in such a framework.
- Describing an approach for extending the execution tracing technique so that it can be integrated into PKI via the supplemental use of certificates.

In Section 2, we explain how utilizing trust in a code security framework is useful in certain situations. Execution tracing is then described briefly and its use in establishing trust levels in such a framework is justified. Section 3 outlines the main entities involved in a security framework that involves an extended version of the original execution tracing technique. The operations of the various entities in this framework is traced in Section 4. Section 5 discusses related work involving certificate use with agents, and Section 6 concludes with a summary and a discussion of future work.

2. TRUST IN A CODE SECURITY FRAMEWORK

In a previous paper [18], we proposed the use of trust as an important component in the development of a framework for mobile agent code security. We briefly summarize the two original propositions for employing trust in a mobile agent security framework:

Trust provides a basis for selecting a combination of code security techniques. A large portion of the techniques developed so far have been found to be deficient in one or more aspects upon closer scrutiny. It is likely that a future security infrastructure that addresses the code security issue comprehensively will need to incorporate a combination of techniques. The level of trust in a particular execution environment can be used as a basis for the appropriate combination of techniques to apply in that environment.

Trust permits more flexible deployment of a code security technique. Another important consideration in the deployment of a security technique is the overhead incurred when using it. For example, executing obfuscated code could result in a performance lag (as contrasted to normal code) that might become intolerable in certain applications. For an agent with a predefined itinerary running on a trusted platform, we could apply obfuscation to selected portions of its code and state that are critical to its correct functioning, while an agent running on an untrusted platform would have its entire code obfuscated.

2.1 Using execution tracing to establish trust

In general, literature surveys [12], [14], [22] on the variety of code security techniques available have classified them into techniques that *prevent* meaningful tampering and techniques that *detect* such tampering. Techniques such as code obfuscation and encrypted functions are designed to prevent meaningful manipulation of the agent code (i.e. manipulation in a manner where the effects of the manipulation are known or predictable to the manipulator and which will eventually lead to the procurement of some advantage to the manipulator) and are correspondingly more complicated in their deployment. However should such manipulation occur, there is no provision to detect it and undertake punitive measures on the trespassing host platform. In effect, the trust model is very simplistic here; no entity is trusted at all and maximal measures are undertaken to prevent any possible security breach.

Techniques that detect tampering, such as execution tracing, allow the development of a more complex trust model. By allowing a host platform to execute an agent and then checking the results of the execution, it becomes possible to identify the different levels of trust that can be established in that host platform through suitable mathematical or intuitive analysis of the results (for example, considering the amount of correct executions as contrasted against incorrect ones). These different levels of trust can then be used to select a combination of code security techniques (in addition to execution tracing) that need to be applied on that particular host platform, in line with the original motivations for the use of trust in a code security framework.

With regards to this, we will employ execution tracing as our core code security technique in the framework that we are about to develop and we describe how some of the original drawbacks of execution tracing can be overcome in our approach. There are other detection mechanisms available such as forward integrity [23] and state appraisal [7]; execution tracing however offers the important advantage of being able to detect tampering of any part of the agent that is actually executed as opposed to only specific portions, as is the case with the former two mechanisms [22].

2.2 Execution tracing

In execution tracing, a host platform executing an agent creates a trace of an agent's execution that contains precisely the lines of code that were executed by the mobile agent as well as all the external values that were read by the mobile agent. When the mobile agent requests to move, a hash of this trace and of the agent's intermediate state are signed by the host platform. This guarantees non-repudiation by providing evidence that a specific state of execution (along with its corresponding results) was achieved on the host platform prior to migration. The trace is then forwarded together with the actual mobile agent to the next host platform on the mobile agent's itinerary. The new host platform will store this trace together with the new trace it creates when executing the mobile agent.

Upon return of the mobile agent, the agent owner may (if she/he suspects that the mobile agent was not correctly executed) request the complete trace of the agent's execution commencing from the first host platform. The agent owner will then simulate the execution of the mobile agent based on the information contained in the trace. This simulation will result in an intermediate state and identify the next host platform to which the mobile agent migrated to. The agent owner requests from this platform the hash of the trace and of the agent's intermediate result, both signed by the first host platform. If these hashes correspond with the trace received from the first host platform and the intermediate state of the simulated execution, the agent owner knows that the first host platform undertook a correct execution of the agent. This process is repeated until the last host platform that the agent migrated to. If at some point a discrepancy is found during the verification of the trace provided by a particular host platform, then a malicious host has been detected.

There have been some criticisms of this approach. The main drawbacks are the size and number of logs to be retained, and the fact that the detection process is triggered only on suspicion that an agent has been manipulated. Other problems include the difficulty of tracing the execution results of multi-threaded agents.

3. COMPONENTS OF THE MOBILE CODE SECURITY FRAMEWORK

In this section, we describe the various entities in our code security framework, including the certificates that will be used to facilitate the integration of the framework with a PKI. The PKI is currently the most widely adopted method for deploying public key cryptography techniques to achieve scalable security (particularly authentication) in a distri/-buted environment. Integration of the framework within a PKI would thus simplify its actual implementation for a real-world mobile agent application scenario.

3.1 Agent templates

Aridor and Lange [2] have suggested the use of agent design patterns to simplify the process of constructing a mobile agent for specific applications. An agent developer can thus select from a 'standard library' implementations of these patterns and combine multiple patterns to compose a complete mobile agent. A possible further development of mobile agent templates would involve refining these standard implementations to fit the specific requirements of a mobile agent application, such as typical e-commerce scenarios described in [9]. In such an instance, the template need only be instantiated with the necessary parameters (such as the identity of the agent, its travel itinerary, the product to be purchased or inquired after, etc) and it could then be deployed immediately.

3.2 Entities in the framework

In our framework, the following entities can be defined and are described below.

3.2.1 Certificate authority (CA)

The CA in our environment corresponds to the CA in a PKI, with similar roles and responsibilities. It will issue the necessary certificates to the other entities in the framework and undertake other necessary key management activities.

3.2.2 Agent owner platform

The agent owner platform is the originating platform from which a mobile agent is initially launched from into a network of platforms. This agent could have been composed here or alternatively, assembled by a code producer (a third party that assembles a working mobile agent from templates that can be instantiated later by the agent owner platform) elsewhere and then transported to the owner platform in a secure manner. In any case, the agent owner platform is assumed to be a representative of the human/organizational deployer of the agent.

3.2.3 Host platform

This refers to the execution environment that the mobile agent operates in once it migrates away from its owner platform. Each host platform will be capable of hosting more than an agent and will be capable of providing the necessary resources for safe interaction between multiple executing agents. An agent owner platform can also be a host platform for other agents launched from different agent owner platforms, if this is required.

3.2.4 Verification server

The verification server is a trusted third party that undertakes the verification of execution traces submitted by host platforms on behalf of the agent owner platform. It functions as a second level CA by issuing capability certificates to host platforms that indicate the capability of a particular host platform in executing specific mobile agent templates correctly.

3.2.5 Capability certificates

These are certificates that associate the identity of a host platform with its capability of correctly executing (i.e. executing in a non-malicious manner) specific agent templates, as represented by their identifiers. The primary difference between a capability certificate and a normal certificate is that the public key normally present is replaced in the capability certificate by a template identifier, which identifies the specific instantiated mobile agent template that can be executed safely by the host platform concerned. The certificates are signed with the private key of the respective verification server and can be verified using the public key of the server, which is distributed separately in a normal certificate (issued by the CA) to interested parties.

Although a verification server is capable of tracing the execution of any agent template, identifiers serve two important purposes:

- It allows verification to be matched to resources available. Different agent templates (as specified by their template identifiers) possess different sizes and levels of complexity in the implementation of their functionality. The amount of time and resources to verify their execution will correspond closely to these factors as well. Operators of the verification servers can thus make decisions on which agent types they are willing to verify on the basis of resources allocation policies at their respective servers.
- It permits specific allocation of accountability. A verification server will have to assume some portion of liability within a legal framework in the event it fails to detect the corruption of an agent by a malicious platform. The consequences of such liabilities would differ

according to agent type; higher penalties would be incurred if the agent in question was meant to engage in a financial transaction then if it were not. Operators can again select which liabilities they are willing to risk on their respective verification servers.

3.2.6 *Execution certificates*

In addition to capability certificates, execution certificates are used to identify the success of such a validation process and are prepared and submitted by a verification server to a host platform upon successful completion of the validation process. Each execution certificate contains a hash of the agent code and state, a timestamp, the identity of the verification server that examined the trace, the identity of the host platform that undertook the execution of the mobile agent as well as the results of validating that trace. Each execution certificate is signed with the private key of the verification server and is validated in the same manner as the capability certificates.

3.2.7 *Capability certificate revocation list*

This list is analogous to the standard certificate revocation list (CRL) found in a PKI, and is used to invalidate previously issued capability certificates in the instance of an invalid execution trace being detected by a verification server. Each entry in this list is submitted by a verification server and will include the identity of the server and the host platform, the fault detected in the trace and a timestamp. An agent can choose to inspect this list prior to migrating to a new platform.

3.3 Changes in the execution tracing protocol

The primary difference between the original execution tracing protocol and the way it is employed in this framework is that the agent owner platform is no longer responsible for verifying the traces submitted by host platforms. Instead a new entity, the *verification server* undertakes this activity on behalf of the agent owner. Based on the results of this trace verification, the verification server will issue certificates to host platforms to indicate their capability of executing mobile agents in a valid manner. These certificates are then distributed onwards to owner platforms with mobile agents that wish to migrate to these host platforms. The agents will then decide on the basis of the information in the certificates, whether it is safe or not to migrate to the respective host platforms on their given itinerary. There will be a large number of verification servers distributed throughout the system, capable of verifying the traces of every single agent executing on all the host platforms in the system.

4. OPERATION OF THE FRAMEWORK

In this section, we describe the operations of the framework with reference to Fig. 1 by following the trail of a mobile agent as it migrates through the network.

4.1 Before migrating to a new host platform

Mobile agents will need to engage in a preliminary interaction with their intended target host platform prior to migrating there from their current host platform (or owner platform). Each agent will have a list of template identifier(s) representing the actual template(s) that they are composed from. This, along with the immutable code portion of the agent, is signed by the agent owner platform to prevent tampering at other host platforms. The identifiers are submitted to the host platform that the mobile agent intends to migrate to (Fig. 1 a).

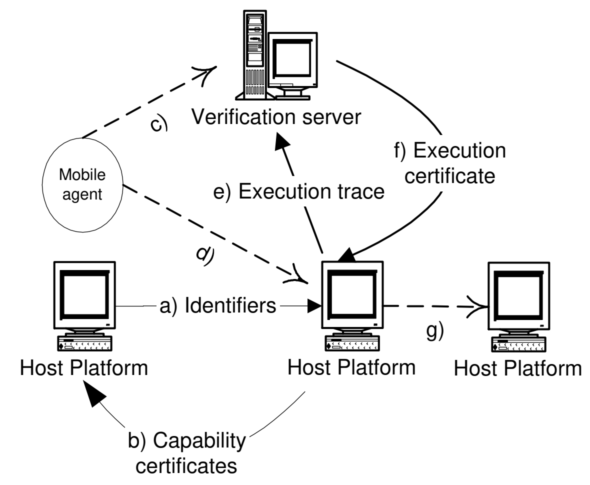

Figure 1: Operation of framework

The host platform receiving the identifiers then checks to see whether it possesses any capability certificates that contain some or all of the identifiers specified. These are then submitted back to the mobile agent, who can then make the decision to migrate or not (Fig. 1 b). As a safety check, the agent may in addition consult the capability certificate revocation list to ascertain whether any of the capability certificates received have been invalidated. If the agent is not satisfied regarding the safety of its intended destination, it will select another host platform either dynamically or proceed to the next destination on a predefined itinerary.

4.2 Migration to and execution on the new host platform

Once a decision is made, the mobile agent migrates over to its intended destination (Fig. 1 d), where execution commences and an execution trace is prepared by the host platform. This trace is submitted to the same verification server (Fig. 1 e), who can validate it accordingly. If this trace verifies properly, an execution certificate is prepared and submitted back to the host platform (Fig. 1 f). A copy of this certificate is stored locally at the host platform and the original is dispatched along with the mobile agent to the next platform (Fig. 1 g). The execution certificate is necessary to ensure acceptance of the mobile agent by the next platform.

If there is a deviation in the trace, which suggests the possibility of malicious tampering, no certificate is issued and instead an entry is updated by the verification server in the capability certificate revocation list to invalidate any previously issued capability certificates. This entry is only applicable for certificates issued by that verification server; it may not invalidate capability certificates issued by other verification servers.

4.3 Periodical checking of traces

Although execution traces are submitted to a specific verification server from a host platform every time an agent is executed on that platform, these traces need not be checked all the time. The idea is to rely on the established trust between the host platform and the verification server as well as the implicit understanding on part of the host platform that violation of this trust relationship, if detected, will entail a greater economic loss (through appropriate sanctions) as contrasted to the possible gain that might be obtained through an undetected violation. In this way, the overhead on each verification server involved in checking every single execution trace can be reduced.

4.4 Necessity of execution certificate at receiving host platform

The reception of the execution certificate is compulsory for the new host platform to accept and execute the mobile agent. If such a certificate is not forthcoming within a predefined time limit, a communication failure or malicious tampering is assumed and the mobile agent is discarded from the new platform it has just migrated to. This provides an important aspect of safeguarding the host platform from mobile agents that may have now become potentially dangerous due to possible subversion from the previous platform.

4.5 Safeguarding against malicious verification servers

To safeguard against the possibility of a malicious verification server (i.e. a verification server that intentionally maligns the host platform by adding invalid entries into the capability certificate revocation list) as well as to provide safety redundancy (in the event of the failure of a verification server that a host platform depends on), a host platform can request capability certificates for the same template from several different verification servers. It can then select to alternate between these verification servers when submitting certificates in response to requests from mobile agents that are about to migrate to it. Copies of execution certificates retained at the host platform from different verification servers can serve as evidence in helping to clarify the possible situation where a malicious verification server attempts to malign the host platform.

4.6 Advantages achieved over original technique

The modification of the original execution tracing protocol to the form that is being utilized in the framework lends several important advantages :

1. Execution tracing can now be performed by any verification server on a host platform once a mobile agent on that platform completes executing. In the original protocol, execution tracing was only possible at the agent owner's site after the mobile agent had completed a tour of its itinerary. In addition, tracing is now compulsory for each host platform and not an optional activity triggered by a suspicious agent owner, as is the case with the original protocol. This allows the detection of malicious tampering as soon as it occurs at any point on the agent's itinerary.

2. Trace logs no longer need be retained at the host platform; maintenance of all tracing evidence is delegated to the verification server. This information will be encapsulated in the form of capability and execution certificates and will be more succinct; being a summary of the results of a trace rather than a log of the entire trace itself.

3. By offloading the verification activity to another entity (the verification server), the agent owner is no longer inflicted with tedious trace checking activities that may become insurmountable when the number of mobile agents launched from it increases or the communication between distant host platforms are unreliable. This increases the scalability of the protocol.

5. RELATED WORK

The integration of a mobile agent system within some form of certificate infrastructure has been the subject of recent work on mobile agent security. In his work on privilege management for mobile agents, Jansen [13] introduces attribute certificates as a flexible manner to express the privileges associated with a mobile agent. Attribute certificates are combined with a mechanism that permits the instantiation of a policy engine in order to provide a framework that can be tailored to meet the security policy of individual applications. Hu [11] in turn proposes several mechanisms for building up an agent-oriented PKI from SPKI/SDSI [5] and X.509 certificate standards. This include the use of trust delegation mechanisms such as chain-ruled, threshold and conditional delegation. Certificates are employed in these approaches as a means to address the first aspect of mobile agent security: host security. In general, a certificate is dispatched along with the mobile agent and is evaluated by the receiving host platform to determine the authorization rights to various system resources that should be accorded to the agent.

Our work is the first to address the use of certificates within a PKI to tackle the alternative aspect of mobile agent security: code security. This is achieved primarily through the use of capability certificates that are submitted by a potential host platform to a mobile agent to indicate the safety of its execution environment for the agent concerned. Execution certificates are also issued at the end of a successful trace verification to indicate to the next platform in the agent's itinerary that a valid execution of the agent was achieved on the current platform.

6. CONCLUSION

In this paper, we have discussed a security framework for mobile agents that extends the execution tracing technique to make it appropriate for large scale environments. The key entity in this framework is the verification server that undertakes the execution tracing activity on behalf of the agent owner platform and issues capability and execution certificates (in a manner analogous to a CA) to the various host platforms in the system. Capability certificates are produced to verify the safety (or otherwise) of the execution environments of various host platforms, while execution certificates are used to provide an indication of the validity of agent execution on a particular host platform.

There remains much work that needs to be done in extending the conceptual framework that we have just described. In particular, a practical implementation of execution tracing has yet to be realized in any major mobile agent system. The interactions between the different entities in the framework needs to be formalized so that specific security properties can be identified and maintained, if possible. Investigation needs to be carried out with regards to the other types of security techniques that can be employed in conjunction with execution tracing and the manner in which they can be integrated into the framework. The detailed format of the capability and execution certificates needs to be examined to see how they might be successfully integrated into the certificate management activities of a standard PKI. Last but not least, the implications of issuing and employing certificates in the manner described here needs to be investigated in greater detail to determine its viability for actual application.

7. ACKNOWLEDGMENTS

This research is funded in part by QinetiQ and EPSRC Magnitude project (reference GR/N35816).

8. REFERENCES

[1] C. Adams and S. Lyold. *Understanding Public Key Infrastructure : Concepts, Standards and Deployment Considerations.* Macmillan Technical Publishing, 1999.

[2] Y. Aridor and D. B. Lange. Agent Design Patterns : Elements of Agent Applications Design. In *Proceedings of the 2nd International Conference on Autonomous Agents*, May 1998.

[3] Cristiano Castelfranchi and Rino Falcone. Principles of Trust for MAS : Cognitive Anatomy, Social Importance, and Quantification. In *Workshop on Deception, Fraud and Trust in Agent Societies : Proceedings of the 3rd International Conference on Multiagent Systems*, 1998.

[4] D. M. Chess. Security issues in mobile code systems. In *Mobile Agents and Security*, number 1419 in LNCS. Springer-Verlag, 1998.

[5] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. Spki certificate theory. In *RFC 2693*, September 1999.

[6] H. S. Nwana et al. Agent-mediated electronic commerce : issues, challenges and some viewpoints. In *Proceedings of the 2rd International Conference on Autonomous Agents*, 1998.

[7] W. Farmer, J. Guttman, and V. Swarup. Security for mobile agents : Authentication and state appraisal. In *European Symposium on Research in Computer Security*, number 1146 in LNCS. Springer-Verlag, 1996.

[8] Li Gong. Java Security Architecture (JDK1.2). Technical report, Sun Microsystems, March 1998.

[9] R. H. Guttman, A. G. Moukas, and P. Maes. Agents as mediators in electronic commerce. *Electronic Markets*, 8(1), May 1998.

[10] Fritz Hohl. Time limited blackbox security: Protecting mobile agents from malicious hosts. In *Mobile Agents and Security*, number 1419 in LNCS. Springer-Verlag, 1998.

[11] Yuh-Jong Hu. Some thoughts on agent trust and delegation. In *Proceedings of the 5th International Conference on Autonomous Agents, Montreal*, June 2001.

[12] Wayne Jansen. Countermeasures for mobile agent security. In *Computer Communications, Special Issue on Advances in Research and Application of Network Security*, November 2000.

[13] Wayne Jansen. A privilege management scheme for mobile agents. In *Workshop on Security of Mobile Multi-Agent Systems : Proceedings of the 5th International Conference on Autonomous Agents*, June 2001.

[14] Neeran Karnik. *Security in Mobile Agent Systems.* PhD thesis, Department of Computer Science and Engineering, University of Minnesota, 1998.

[15] Danny B. Lange and Mitsuru Oshima. Seven good reasons for mobile agents. *Communications of the ACM*, 42(3), 1999.

[16] G. Necula and P. Lee. Safe kernel extensions without run-time checking. In *Proceedings of the 2nd Symposium on Operating System Design and Implementation (OSDI '96), Washington*, October 1996.

[17] T. Sander and C. F. Tschudin. Protecting mobile agents against malicious hosts. In *Mobile Agents and Security*, number 1419 in LNCS. Springer-Verlag, 1998.

[18] H. K. Tan and L. Moreau. Trust relationships in a mobile agent system. In *5th IEEE International Conference on Mobile Agents, Georgia, USA*, December 2001.

[19] Giovanni Vigna. Cryptographic traces for mobile agents. In *Mobile Agents and Security*, number 1419 in LNCS. Springer-Verlag, 1998.

[20] D. Volpano and G. Smith. Language issues in mobile program security. In *Mobile Agents and Security*, number 1419 in LNCS. Springer-Verlag, 1998.

[21] U. G. Wilhelm, S. Staamann, and L. Buttyan. Introducing trusted third parties to the mobile agent paradigm. In *Secure Internet Programming : Security Issues for Mobile and Distributed Objects*, number 1603 in LNCS. Springer-Verlag, 1999.

[22] Uwe G. Wilhelm. *A Technical Approach to Privacy based on Mobile Agents Protected by Tamper-resistant Hardware.* PhD thesis, Ecole Polytechnique Federale de Lausanne, 1999.

[23] Bennet S. Yee. A sanctuary for mobile agents. In *Secure Internet Programming : Security Issues for Mobile and Distributed Objects*, number 1603 in LNCS. Springer-Verlag, 1999.