(How) Can Mobile Agents Do Secure Electronic Transactions on Untrusted Hosts? A Survey of the Security Issues and the Current Solutions

JORIS CLAESSENS, BART PRENEEL, and JOOS VANDEWALLE
Katholieke Universiteit Leuven—ESAT/SCD-COSIC

This article investigates if and how mobile agents can execute secure electronic transactions on untrusted hosts. An overview of the security issues of mobile agents is first given. The problem of untrusted (i.e., potentially malicious) hosts is one of these issues, and appears to be the most difficult to solve. The current approaches to counter this problem are evaluated, and their relevance for secure electronic transactions is discussed. In particular, a state-of-the-art survey of mobile agent-based secure electronic transactions is presented.

Categories and Subject Descriptors: A.1 [**Introductory and Survey**]; E.3 [**Data Encryption**]; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms: Security

Additional Key Words and Phrases: Mobile agent security, electronic transactions, malicious hosts

1. INTRODUCTION

Business on the Internet has now become standard practice. Books, music, computers, and so on, are bought electronically by consumers. Companies do (part of) their business with other companies in an electronic way. Security and cryptography are important enablers: without them, electronic commerce would not work in the real world; moreover, they are the core technical means to implement secure electronic payments and transactions.

The most frequently used electronic payment mechanism on the Internet today is probably transferring credit card information over a secured (SSL/TLS [Dierks and Allen 1999]) connection from a customer's browser to a

J. Claessens, at the time this article was written, was funded by a research grant of the Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT). This work was also supported in part by the FWO-Vlaanderen project G.0358.99 on Open Security Systems for Agent-Based Applications (OSABA), and by the Concerted Research Action (GOA) Mefisto-2000/06 of the Flemish government.

Authors' address: Katholieke Universiteit Leuven, Dept. of Electrical Engineering—ESAT/SCD-COSIC, Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium; email: URL: http://www.esat.kuleuven.ac.be/cosic/; email: joris.claessens@esat.kuleuven.ac.be.

merchant's Web server. This is, however, far from an ideal system: the credit card information is only protected in transit; the information should remain secret, yet it is communicated to the merchants and then mostly stored in their computer systems, vulnerable to attackers. Secure Electronic Transaction (SET) [SET]—note that this article's title refers to the general concept of secure electronic payment and transaction mechanisms and not to this particular protocol—is a credit card payment system which is much more secure, as the credit card number is cryptographically protected from the merchant, and as the payment token is not static but is each time cryptographically linked (using a digital signature) to, among other things, the payment amount. An alternative to credit card payments is the electronic form of cash. This area has been researched intensively (an overview can be found in O'Mahony et al. [2001]), and systems with various properties have been proposed: online versus offline, unconditional or revocable anonymity, and so on. While electronic cash works with individual electronic coins, other systems are based on electronic checks and a balance. Micropayments are efficient systems intended for very small payments, for example, deployed in pay-per-click Web applications. Finally, while digital signatures are often the cryptographic basis behind a particular payment system, they can also be used to directly sign a message (e.g., secure email), a contract, or a business transaction (e.g., Signed XML [Eastlake et al. 2002]).

In the traditional way of "Internet-ing," a user runs client programs on her local machine. Clients send requests to server programs, get responses back, possibly process these responses, and show the results to the user. The user then has to interpret these results, make decisions based on this interpretation, and start the process again. In some applications this is becoming an increasingly time-consuming and difficult task for end-users. First, there is an increasing amount of available information on an increasing number of servers. When users want to buy a certain item at the lowest price, they have to consult the numerous sites that sell this item, and manually compare the prices. Second, some applications require a lot of interactivity, for example, querying a public database or electronic auctions. Finally, the Internet is expanding to mobile devices, which still have a slower and more costly connection and, more important, are usually not continuously online. Mobile devices also have limited power consumption. Web crawlers and robots address the first two issues to some extent. However, they run on the user's machine and require a continuous active connection to the network.

Without giving a formal definition, a software agent is a very generic term for a piece of software that can operate autonomously, and that helps facilitate a certain task. Software agents can communicate, they can be intelligent, and often have learning capabilities. Mobile software agents are agents that can travel from one computer to another computer. They are sent by end-users and visit a series of hosts. The mobile agents are executed locally on these hosts to perform their tasks, and will return to the end-users to report their results. It seems clear that the mobile agent paradigm offers a potential answer with respect to the observations made above. Chess et al. [1997] conclude that "while the individual advantages of agents do not represent an overwhelming motivation for their adoption, the creation of a pervasive agent framework facilitates a very large number of network services and applications." Other researchers shared this vision, acknowledged the advantages of mobile agents, and predicted the deployment of mobile agents on the Internet "*within the next few years*," see Kotz and Gray [1999] and Lange and Oshima [1999].

There are, however, some security issues concerning mobile software agents. These will have to be tackled before the agent concept can be used for electronic commerce purposes. In particular, if mobile agents would be capable of making secure payments, they would not need to return to the end-user each time such a payment were needed. A secure electronic payment or transaction always involves the use of secret information/keys, either directly (e.g., a credit card number or simple electronic coins), or by mathematically producing a payment or transaction means using these secret data (e.g., an electronic check or a digital signature on a contract). Protecting these secret data is a very difficult problem. The "standard" end-user PC operating system has been proven to be very vulnerable; in particular, the cryptographic keys of applications can often be located on the hard disk, or in memory, as shown by Shamir and van Someren [1999]. Even hardware, such as smartcards, is not always able to protect secret data against advanced attacks; see, for example, Kocher et al. [1999]. Thus it seems to be very difficult, if not impossible, to protect secret data contained in mobile software agents from malicious hosts.

1.1 Scope and Outline of This Article

This article investigates if and how mobile agents can conduct secure electronic transactions on untrusted hosts. The article positions this challenging problem in the overall framework of mobile agent security issues. Section 2 gives an overview of the different security issues concerning mobile agents. Section 3 then discusses the issue of untrusted hosts, which is the agent security issue most relevant to our problem. This section presents solutions that protect entire mobile agents from malicious hosts. More pragmatic solutions are discussed in Section 4. This section deals with some solutions that do not necessarily protect mobile agents from malicious hosts, but allow these mobile agents to perform electronic transactions securely or with limited risk. Further discussion is given in Section 5. This section also addresses the question of *why* mobile agents should try to perform secure electronic transactions on untrusted hosts. Finally, Section 6 concludes the article.

The purpose of this article is primarily to present a survey on the state of the art of mobile agents and secure electronic transactions on untrusted hosts; in addition, ideas, open issues, and interesting directions for the future are indicated. The reader is not expected to have a thorough knowledge of security and cryptography, although we assume familiarity with some basic concepts, such as digital signatures and encryption. This survey is not intended as a recipe guide for actually building electronic transaction systems based on mobile agents. It rather constitutes a reference for people who want to learn about the problem and the current solutions. It is therefore aimed at researchers interested or active in this field, but also at system architects who want to become familiar with mobile agent security issues and the potential solutions to tackle the relevant problems.

2. MOBILE AGENTS AND SECURITY

Security is without doubt one of the cornerstone issues for mobile agents [Chess et al. 1995]. Why does the mobile agent concept raise (new) security issues at all? Chess [1998] states that it does so because it violates a number of assumptions that underlie most existing computer security measures:

— computer programs take actions on behalf of a person who can easily be identified, and who intends these actions to be taken;
— computer programs are obtained from easily identifiable and generally trusted sources (thus Trojan horses, programs that attack a system by doing something else than what the user intends/expects, are rare);
— security threats come from attackers running programs with malicious intent, therefore security measures should authenticate the user, and make sure that programs run by this user can only do those things that the user is allowed to do;
— computer programs do not move from one system to another, unless users intentionally transmit them.

The last assumption is especially very questionable in today's computing environment. The "Internet Worm" was released in November 1988; see Spafford [1988]. It exploited bugs only in certain network services, and was not mobile because explicitly allowed by the platforms. However, without always being explicitly labeled in this way, today's computer systems already incorporate a lot of mobility features: emails can contain program code that is automatically executed when opening the email; Web pages can contain Java applets that are executed when viewing the page; Word documents can contain macrocode; computer programs from unknown sources are run by users without any hesitation, and so on. The aforementioned assumptions have thus certainly become questionable in today's computing environment and, unfortunately, related security problems have already resulted. It is clear that for fully featured mobile agent environments these assumptions are completely violated, thus raising new security issues. Different security aspects of mobile agents can be identified, and have already been studied in the literature [Chess 1998; Farmer et al. 1996a; Hassler 2000; Jansen and Karygiannis 1999; Jansen 2000; Loureiro 2001; Tschudin 1999].

Insecure Networks. Mobile agent systems are deployed over standard computer networks. The basic network security aspects [Bellovin 1989] are obviously a mobile agent security issue too. Mobile agents should be protected while they are in transit from one host to another host. The communication between agents and users, and between agents themselves, should also be protected. Agents, hosts, and users, as well as nonparticipating entities, could potentially eavesdrop or tamper with the communication, or impersonate participating entities.

Thus the typical *cryptographic security services*—entity authentication, data authentication (data origin authentication and data integrity), and data confidentiality—should be provided. The communication channels should therefore be cryptographically secured. Standard mechanisms are available for this: SSL/TLS [Dierks and Allen 1999] at the transport layer, or IPsec [Doraswamy and Harkins 1999] at the network layer. An easy solution is for agent platforms to provide this as a service to the agents [Claessens et al. 2001]. Alternatively, it can be established by the agents themselves.

The protection of the cryptographic keys that are needed to secure the communications is very important and is clearly dependent on the other mobile agent security issues. Malicious agents should be prevented from stealing a host's private keys. If agents carry their own keys, they should be able to protect them from malicious hosts. The authentication of both agents and hosts [Berkovits et al. 1998] is not only an important aspect of the protection of the communication, it is also very relevant with respect to the malicious agents/hosts issue. Authentication is the first step in the process of determining whether an agent/host should be trusted, and thus whether, respectively, the host should execute the agent, or the agent should migrate to that host.

Malicious Agents. In an open mobile agent system, it is expected that there will be mobile agents with malicious intentions. Other agents and agent platforms should be protected from these malicious agents.

As mentioned above, agent *authentication* allows a host to identify an agent, or in practice the signer of an agent (which can be the author, the owner, or the sender) [Gong and Schemers 1998]. Agents should run in a sandbox environment in which they have limited privileges, in which they are safely interpreted [Volpano and Smith 1998], and in which they are also protected from one another. This is exactly how Java applets are executed in a browser. Sandboxing for mobile agents can thus be provided by Java directly. Thus, to some extent, the Java security model already provides a partial solution for the malicious agents problem. Stronger protection, including resource control, can be built on top of the Java environment; see, for example, Binder [1999]. There are also other programming languages providing safe execution, such as Safe-Tcl [Ousterhout et al. 1998]. Agents can request extra privileges, which are granted by the host if it trusts the agent's signer for these particular operations. Note again that Java applets can do the same thing. With an *authorization* language, a complete security policy can be implemented on a host, as described in Karjoth et al. [1998], specifying which agents (identified by the signer of the agent) are allowed to do what (sensitive) operations. In addition, resource usage control, or *allocation*, should prevent agents from, for example, flooding a host and denying resources to other agents.

In addition to authentication, agents can *carry a proof* that enables a host to determine whether the agent's code is safe to install and execute; see Necula and Lee [1998]. In particular, an easily checkable proof is attached to the code that the agent's execution does not violate the security policy of the receiving system. Not only the code, but also the agent's state is relevant for security. With *state appraisal* (see Farmer et al. [1996b]), the set of privileges an agent needs is computed depending on the state of the agent. Dangerous state modification can be detected. In some sense this protects against a malicious host that corrupts an agent's state before it migrates to another host.

Finally note that Gray et al. [1998] observe that groups of hosts should also be protected against malicious agents. This counters low-profile attacks (e.g., resource consuming) that are not detected by individual hosts.

Malicious Hosts. Although the problem of malicious agents seems more or less easy to solve, protecting agents against malicous hosts seems a very difficult, and even impossible, task. To demonstrate the problem of malicious hosts, Hohl [1998a] presented a model in which the following attacks by malicious hosts were identified: spying out an agent's code, data, and control flow; manipulation of an agent's code, data, and control flow; masquerading of the host; denial of execution (note that there is also reexecution) of an agent; spying out interaction with other agents; returning wrong results of system calls issued by the agent. An agent is completely under the control of a host, and a malicious host can therefore do almost anything.

If and how mobile agents can do secure electronic transactions in these circumstances is the question that is investigated in the following sections. It is important to notice that it is impossible to prevent the actual attacks themselves. However, the purpose of the solutions that are discussed in the remainder of the article is to render these attacks useless, or at least to allow for detection. A comparison can be made in the area of network communications: encryption does not prevent an attacker from eavesdropping on communications, but it does make communications incomprehensible.

Malicious Users. It should not be forgotten that users are the root of most security problems. Besides eavesdropping on and tampering with the communication, creating and sending out malicious agents, and running malicious platforms, users can exhibit other undesirable behavior. With respect to electronic transactions, for example, users could deny having sent an agent, or could refuse to honor payments made by an agent. Nonrepudiation and auditing services should therefore be provided. Signing an agent once is not enough, as this only proves who the owner is but not that this owner intended to send it to a host at that particular moment. A legal framework should determine to what extent a user remains responsible for tasks that were delegated to an agent.

3. SOLUTIONS TO THE MALICIOUS HOSTS PROBLEM

This section presents general solutions to the malicious hosts problem and discusses their relevance for secure electronic transactions. Remember that a mobile agent consists of code, data, and a state. We start with some solutions that try to avoid the problem of malicious hosts in the first place. Thereafter, solutions that protect the data part of an agent are discussed. Finally, we look at solutions that protect the integrity of an agent's execution, as well as the privacy of an agent's execution. The execution of an agent is closely related to the code and the state.

3.1 Avoiding the Problem

Just as humans in the real world avoid interaction with organizations or humans with a bad reputation, social control and *reputation* can also be used in the mobile agent world as a security mechanism. Rasmusson and Jansson [1996] claim that this "soft security" approach, in which the participants themselves are responsible for the security, is more robust than the usual "hard security" approach, in which security is enforced by technical, yet often bypassable, means and/or via a trusted external party. However, identifying the other party by an authentication mechanism remains an important technical prerequisite before its reputation can be considered.

Whether based on reputation or otherwise, mobile agents can only visit a limited set of (authenticated) hosts that they (their user) trust(s). A closed group of trusted execution platforms can thus be established. Note that if two mobile agents do not have a trusted host in common, yet want to interact, they can always travel to those trusted hosts that are in proximity to one another (with respect to available bandwidth); this set of hosts can be called a *virtual trusted third party* (see de Decker et al. [2000]).

Trusted hardware can be attached locally to each host. Sensitive code can be sent encrypted to these *trusted environments*. These execution environments—and especially the cryptographic keys that they contain—cannot be controlled or tampered with by the hosts (or their operators). Examples of trusted hardware are secure coprocessors as proposed by Yee [1999], and smart cards as proposed by Loureiro and Molva [2000]. Wilhelm et al. [1998] present another system using tamperproof environments, and include a nice discussion on the notion of trust. The trusted hardware approach again constitutes a closed group of trusted execution platforms. Trust is, however, not established by legal or social means, but provided by (costly) technical measures.

Secure Electronic Transactions. It is clear that avoiding the problem of malicious hosts as described in the previous paragraphs is a "solution" to enabling mobile agents to perform secure electronic transactions. It seems, however, either too complicated from a social or legal point of view or, in the case of trusted hardware, just too costly for many applications. The goal of the remainder of this article is to investigate whether there are solutions that require only a minimal level of trust.

3.2 Protecting Data

When mobile agents visit a sequence of hosts, it is desirable that the information that they gather is protected from attackers, including subsequent potentially malicious hosts to which the agents will migrate. Karjoth et al. [1998] define a number of security properties that should be achieved with respect to an attacker that has captured a mobile agent holding a *chain of encapsulated information*: data confidentiality, so that only the originator can extract the information; nonrepudiability, so that hosts cannot repudiate information (e.g., in the case of an offer); forward privacy, so that the identities of the previous hosts are not disclosed; strong forward integrity, so that no gathered information can later be modified (except optionally for updates by the providing hosts themselves [Loureiro et al. 1999]); publicly verifiable integrity, so that anyone, including hosts and the agent itself, can verify the chain of information; insertion resilience, so that no data can be inserted unless explicitly allowed; and truncation resilience, so that a chain of information can only be truncated by a host that colludes with the attacker. A number of solutions and tools for partial result encapsulation and chaining have been proposed that offer (a subset of) these properties [Karjoth et al. 1998; Loureiro et al. 1999; Yee 1999; Young and Yung 1997]. These solutions are usually based on the normal cryptographic primitives: hosts digitally sign the data that they give to the agent, and the data are encrypted with the public key of the agent owner; the "chaining" property is, for example, provided by including the hash of the previously gathered data and the identity of the next host into the current data before signing them. Note that flaws have been identified in some of the proposed protocols [Roth 2001].

Instead of cryptographically preventing attackers from tampering with agent data, it might be sufficient in some applications to detect tampering in some other way. For example, dummy data items can be inserted by the agent owner that will not be modified by legitimate hosts. If these "detection objects" (see Meadows [1997]) have not been modified, then probably the real data have also not been corrupted.

Secure Electronic Transactions. Although these solutions for protecting agent data are useful and necessary for protecting the context of an electronic transaction (the offer, the agreement, etc.), they are not helpful for the secure electronic transaction itself. When confidentiality is provided, only the originator of the agent can decrypt the information, and not the agent itself, because in that case also the host can obviously have access to it. For electronic transactions the agent itself would need access to, for example, a private key. It is clear that in this case protection of data is not enough because we have to protect the data with which data are protected, and so on. Thus, we can conclude that some form of execution privacy is also required.

3.3 Execution Integrity

Execution integrity is about ensuring the correct execution of an agent. The protection of code and state of an agent is important here. Obviously, code and state should first be protected from outside attackers by, for example, digitally signing it and, then encrypting it with the public key of the recipient host. This makes sure that the host receives the correct code and state (optionally, the host should even send a signed receipt). With the solutions discussed here, it will then be possible to check whether this recipient host has executed the agent correctly. Note that these solutions do not prevent a host from, for example, executing an agent multiple times with different inputs.

State appraisal was already mentioned in the section on malicious agents. With state appraisal, invariants can be expressed that an agent's state must satisfy. It can therefore detect tampering to some extent. *Cryptographic traces*, proposed by Vigna [1997, 1998], are detailed and digitally signed logs of the operations performed by an agent during its lifetime. They allow an agent owner to check, after agent termination, whether the execution history of the agent conforms to correct execution, and whether hosts have tampered with code or state. However, complete execution traces are costly to transmit. Moreover, they are costly to verify, because the agent owner has to perform the execution again by himself. Note that only a signed hash of the log is therefore sent, and that an agent owner can request a complete trace afterwards if he does not trust the host. Shorter *proofs of correctness* based on holographic proofs have been proposed by Yee [1999]. These proofs have the property that only a few randomly chosen bits have to be verified. The complete proof must be transmitted, or the proof must be committed to by using a hash tree, which saves bandwidth at the cost of extra communication rounds. Biehl et al. [1998] deploy private information retrieval techniques to improve this and remove some communication rounds: the agent owner is able to receive only the randomly chosen bits of the proof, without revealing to the host exactly which bits. Private information retrieval and the solutions based on proofs of correctness unfortunately remain very costly.

An alternative approach originating from the fault-tolerance world is *replicating agents*. Multiple agents with the same functionality are sent to different hosts. If only some of them are tampered with, while the majority are correctly executed, voting can ensure survival of malicious hosts, as discussed by Minsky et al. [1996].

Hohl [2000] generalizes these solutions using the concept of *reference states*. State appraisal, execution traces, and replication are each a means to detect a difference in behavior between an attacking host and a nonattacking or reference host.

Secure Electronic Transactions. Execution integrity seems to be less relevant with respect to secure electronic transactions, as it does not provide confidentiality. Although the agent's private information is not protected from the host, it might still be possible to discourage illegitimate use of it. For example, a transaction that is digitally signed by a mobile agent might only be taken into account if there is a proof that the agent was properly executed, and if this proof and the digital signature can be linked together.

The concept of replicating agents, and having them executed on different hosts, suggests the following idea: a *k-out-of-n threshold scheme* (e.g., Shoup [2000]) could be deployed to perform electronic transactions. The transaction secret is distributed over n agents that are sent to different hosts. Confidentiality of the private information is provided because none of the agents has the complete secret. It is assumed that there are less than k agents running on malicious and possibly colluding hosts. At least k out of n cooperating agents are needed to complete a transaction. If one agent requests a transaction, the other agents could only cooperate if the untrusted host that is executing this agent shows some proof of correct execution.

Note that this idea is not a solution for execution integrity but is actually an example of the concept of *secure distributed computing* (SDC; see, for example, Cramer [1999]). With SDC, parties can jointly calculate the result of a particular function without revealing the original inputs. Threshold schemes and voting are proposed by Borselius et al. [2001a] as one of the two practical solutions for the malicious hosts problem (the other practical solution in Borselius et al. [2001a] is dependent on one trusted host that makes decisions and from which agents are sent to other untrusted hosts). Another example is presented by Das and Gongxuan [2001] in which multiple mobile agents carry encrypted electronic cash and a share of the key to decrypt the electronic cash.

3.4 Execution Privacy

Although execution integrity provides correct execution of an agent, it does not necessarily keep the code and state of an agent private.

With environmental security measures, the execution of an agent is actually not kept private, but it is only performed when certain environmental conditions are met. *Environmental key generation* [Riordan and Schneier 1998] is a concept in which cryptographic keys are constructed from certain environmental data. For example, an agent or part of it could be encrypted with such a key in order that it would only be decrypted and executed if this environmental data were present at the host. In theory, this could prevent agents from being executed on a malicious host, provided that the environmental conditions that identify whether a host is malicious can be defined.

Code obfuscation can be used to hide the intention of an agent. Although obfuscation might be, theoretically, impossible for some functions, as proven by Barak et al. [2001], practically, execution privacy could at least be provided for the limited amount of time that is required by a malicious host to figure out exactly what is being done by the agent, as suggested by Hohl [1998b]. The intention of an agent cannot only be hidden by adding "mess-up code," but it can also be spread out among a group of collaborating agents, some of which could be bogus agents; see Ng and Cheung [1999].

Function hiding [Cachin et al. 2000; Loureiro and Molva 1999; Sander and Tschudin 1998a] is a cryptographic way to provide execution privacy. It is also referred to as noninteractive, or one-round, secure computing. The general model, applied to our case, is the following. The user, Alice, has a function f, and the untrusted host, Bob, has data x; Alice wants to send a mobile agent to Bob to compute $f(x)$; however, Alice does not want to reveal f to Bob (note that in the general model Bob does not want to reveal x to Alice either; this is not necessarily a requirement here, although sending x to Alice is obviously not an option in the case of mobile agents). Alice encrypts f, and sends $E(f)$ to Bob. Bob calculates $(E(f))(x)$, and sends this back to Alice. Alice calculates $f(x)$ using the key with which she encrypted f. An example application of function hiding is a Secret Query Database [Neven et al. 2000].

Algesheimer et al. [2001] state that the approach based on computing with encrypted functions only works if nobody but the user learns the result of the computation. If the host can observe the output, then it can simply run the code again with different input. The host would then be able to learn the functionality of the encrypted code by simply trying many different inputs and looking at the outputs. Moreover, the host could choose and send back the most suitable output. If the host is to receive some output of the computation then providing execution privacy requires minimal trust in a third party (or in a tamper-resistant module at the agent platform). They present a solution in which a generic independent online entity, a secure computation service, performs some operations on behalf of the mobile agent but does not learn anything about the encrypted computation. This universal solution is intended for "active" agents, agents that visit more than one host before they return to the user; when the agent travels from one host to another then the secure computation service is invoked.

This server-based solution is conceptually the software equivalent to using a trusted hardware module in each host, as discussed in Section 3.1. Note that in the solution of Loureiro and Molva [2000] the "code owner functionalities" (integrity verification and cleartext result retrieval) are delegated to tamperproof hardware available at the remote host (instead of interacting with an online server).

Execution privacy (and integrity) is also a desirable feature in many other software applications, and is not just restricted to mobile agents, for example, computer games, applications containing proprietary algorithms, and viewers for copyrighted content. These programs are installed on the user's computer and are totally under the control of the user, who often has—from the point of view of the distributor—"malicious" intentions. Solutions for tamper-resistant software are thus being developed; see, for example, Aucsmith [1996]. These solutions are also useful in the special case of mobile agents. The solutions often rely on trusted hardware for sensitive operations (such as decrypting program code) and on integrity checking modules.

Secure Electronic Transactions. The solutions for providing execution privacy are obviously very relevant for secure electronic transactions performed by mobile agents.

Environmental key generation appears to be hard to accomplish in practice and probably requires some additional obfuscation, for example, to hide which environmental conditions are tested. One problem is that a malicious host can provide the agent with false information, so it is difficult for the agent to evaluate the real environmental conditions. Note that trusted hardware components (see the TCPA [TCPA] effort) might be helpful here. Such components could be a trustworthy source of information for the mobile agent.

Code obfuscation seems only to be useful for functions that should be hidden for a limited time. As such, it is interesting to deploy it in combination with an electronic transaction application in which the secret is also only valid during a limited timeframe.

With respect to the function hiding concept, for secure electronic transactions f could, for example, be a digital signature function with a hardcoded private key. In addition to protecting the private key itself, the untrusted host should, of course, be prevented from abusing the hidden function and generating signatures without knowing the key. The notion of "undetachable signatures" is therefore introduced by Sander and Tschudin [1998b, c]. Consider the signature routine s and the function f that produces the output to be signed. These two functions should be glued together. The idea is to have $f_{signed} = s \circ f$. The mobile agent has f and f_{signed}. The host can calculate $f(x)$ and $f_{signed}(x)$. The host should only be able to construct $s(n)$ with $n = f(x)$, but should be unable to construct $s(m)$ for an arbitrary m. A secure construction of this idea is presented in Kotzanikolaou et al. [2000] and is discussed in the next section.

4. SECURE ELECTRONIC TRANSACTIONS ON UNTRUSTED HOSTS

The previous section discussed solutions to the general problem of malicious hosts. These solutions were, to an extent, relevant to the specific problem of secure electronic transactions on untrusted hosts. This section discusses some explicit solutions for secure electronic transactions.

4.1 Avoiding the Problem

The problem of malicious hosts can be avoided by simply not sending mobile agents to untrusted hosts, as mentioned earlier. However, in practice and in an open environment this might not be realistic. Obviously, mobile agents can still be safely sent to untrusted hosts if they do not include payment capability.

Kotzanikolaou et al. [1999] propose a system in which there is one master agent and multiple slave agents. The master agent does not travel. The slave agents are mobile but they each travel only to one particular host to negotiate, and they cannot complete a transaction. They return to the master agent with purchase contracts signed by the hosts. The master agent evaluates the signed contracts and presents the results to the user.

Note that even if the mobile agents do not have payment capability, there are still potential malicious host problems. For example, an agent visiting multiple hosts is still vulnerable to a malicious host that corrupts the information gathered at other hosts. The solutions in the previous section are thus certainly still valuable.

4.2 Minimizing the Risk

Just as parents do not usually send their children with a credit card to a shop, but instead with a small wallet containing little more than the estimated amount of money to be paid, users could only entrust their mobile agents with *limited payment capability*. In the case of an electronic cash scheme or micropayment hash-chain-based coins, this is quite easy to achieve: whatever the payment scheme, the mobile agent should just be given a limited amount of electronic coins. If digital signature functionality is required, some measures to limit the use of the private key are needed.

Romão et al. [1999] propose *proxy certificates*. Instead of giving the mobile agent direct access to the user's private digital signature key, a new key pair is generated for the mobile agent. The key pair is certified by the user, thereby binding the user to that key pair, hence proxy certificate, and as such to the transactions that the mobile agent will perform. The lifetime of the certificate is short and therefore revocation is not needed. It should be difficult for a malicious host to discover the private key before the certificate expires (this technique therefore benefits from code obfuscation or agent replication). In addition, the proxy certificate can contain constraints that prevent the private key from being used for arbitrary transactions, in case it is discovered too quickly. Note that the idea of limited delegation of rights to proxies has already been proposed earlier in other security areas (e.g., Neuman [1993]).

Ferreira and Dahab [2001] present an idea in which the private signature key is blinded. A blinded signature can be produced using this *blinded signature key*. The blinding is claimed to be performed in such a way that only the resulting signature can be unblinded, but not the key. Mobile agents carry the blinded signature key and a signed policy that defines the restrictions under which the signature key may be used. The blinding factor can be given to a third party or to the mobile agent. In the first case, the private key is cryptographically protected, as opposed to merely being obfuscated or distributed over multiple agents. The second case corresponds to the regular proxy certificate situation; that is, the host is able to obtain signatures on any message, but the signed policy will still determine which signatures should be considered valid.

In a normal digital signature scheme, a private key can be used indefinitely unless the corresponding certificate has expired or the key has been revoked. Some special schemes with interesting properties in the context of secure electronic transactions for mobile agents are discussed in the following paragraphs.

Yi et al. [2000] propose a digital signature scheme in which users have a long-term key pair, but in which a message-dependent virtually certified *one-time key pair* is generated for each message that has to be signed. A private key that can only be used once would be an ideal solution for a mobile agent (provided the malicious host is not the first to use it). The private key in this system is unfortunately message-related, which makes it unusable for a mobile agent that does not know the message to be signed in advance (note that the message-dependency is the reason why it is a one-time key pair).

A digital signature scheme in which the (base) public key is fixed but the private key is *updated* at regular intervals is described by Bellare and Miner [1999] and Krawczyk [2000]. The key update provides forward security: compromise of the current private key does not enable an attacker to forge signatures from the past. This would be an interesting scheme for mobile agent-based electronic transactions. The public key is fixed, which is an advantage for all verifying entities. When a malicious host compromises a private key, previous keys are not exposed. Revocation is, however, still necessary, as the following private keys will be compromised too. The key update in Bellare and Miner [1999] and Krawczyk [2000] is based on $key_{next} = h(key_{current})$, with h a one-way function, giving the forward security property (note that h is not just a hash function): it is infeasible for a malicious host to find out the previous private keys based on the current key it has compromised. If something like $key_{current} = h(key_{next})$ was used then exposure of the current key would not compromise the next keys. This is a desirable property if agent platforms are untrusted by default, that is, it is assumed that the current private key will be compromised anyway. On the other hand, (expected) exposure of the current key will reveal the previous keys, so usage of these keys should automatically be prevented (e.g., through short-lifetime certificates, or via an online server; see also Section 4.4).

A peculiar concept is introduced by Goldreich et al. [1998]. A user has a primary (long-term) key with associated rights that should not be delegated to others. However, to avoid having to store this primary secret on a vulnerable laptop while traveling, the user may wish to delegate these rights to secondary (short-term) keys. To discourage an excess of delegation, the secondary keys are related to the primary key, so that a certain number of them reveal the long-term key. For mobile agents and secure electronic transactions, this *controlled propagation* seems undesirable; delegation of keys remains an interesting mechanism though.

4.3 Restricting the Use

While the solutions in the previous subsection are based on the fact that the mobile agent has a limited amount of money, or a limited amount of time to perform transactions, the following solutions restrict the use of the mobile agent's payment means. A first example is actually the proxy certificates of the previous subsection, because they can contain constraints, for example, the maximum amount of money that an agent can spend at a particular host.

An *executable cash* scheme, X-Cash, is developed by Jakobsson et al. [1998]. In this scheme the user, Alice, constructs the payment means Ω for the mobile agent as $\Omega = \sigma_{SK_A}(\omega, P), C$. Here, ω is a piece of code that will evaluate a bid (i.e., based on a suitable representation of the bid as input, the outcome of ω will be the amount of money Alice is prepared to pay for the bid), P is the policy, and C Alice's certificate; ω and P are digitally signed by Alice. To make a bid, the host, Bob, creates a suitable signed representation R of his bid, and submits it to Alice's financial institution together with Ω. The financial institution will check whether there is still enough value associated with C and if the transaction is compliant with P; if so, it will transfer the funds as calculated by ω to Bob's financial institution. Note that ω and P are signed so that the host cannot tamper with the transaction. However, this system does not prevent the host from trying out several bids before actually submitting one.

An RSA implementation of the concept of *undetachable signatures* (see previous section) is presented by Kotzanikolaou et al. [2000]. Let $h = \text{hash}(C, req_C)$, with C the identity of the customer (the owner of the mobile agent), and req_C the customer's requirements and constraints. Let $f(.) = h^{(.)} \bmod n$, be the function that produces the output to be signed. Glued together with the signature function, $k = h^d \bmod n$ (d is the private key), gives $f_{signed}(.) = k^{(.)} \bmod n$. Let $x = \text{hash}(S, C, bid_S)$, with S the identity of the server (the host), and bid_S the server's bid. The transaction will be $m = f(x)$, and $s(m) = f_{signed}(x)$. In other words, the mobile agent actually already carries a message-signature pair $r_C, s(r_C)$; from this, it is easy to calculate another message-signature pair, containing the host's bid, but also the user's constraints $r_C{}^{x_S}, s(r_C{}^{x_S})$; the host cannot generate arbitrary message-signature pairs. Borselius et al. [2001b] integrate this technique and the previously discussed threshold signatures into the concept of *undetachable threshold signatures*: multiple agents are required to calculate a signature, which is still subject to constraints.

An undetachable signature scheme allows an agent platform to request a mobile agent to generate a signature on a message of restricted form. The identity of the agent platform is, however, not cryptographically verifiable from the resulting signature. Kim et al. [2001] therefore propose the use of a *proxy signature* for mobile agents. With a proxy signature both mobile agent and agent platform sign a message: in essence, based on an initial signature by the agent owner on some information, the agent platform generates a proxy key pair, and signs the message with this pair; to check the resulting signature both the signature of the agent platform itself and the initial signature of the agent owner should be verified. Kim et al. [2001] extend their scheme to a *one-time proxy signature*; that is, the agent platform is able to sign one and only one message (i.e., if it signs more than one message, its private keys will be revealed). If the identity of the agent platform does not have to be known beforehand (as typically is the case in mobile agent applications), then the proxy signature is *nondesignated*; see Lee et al. [2001].

4.4 Server-Aided Computation

Two cooperative solutions have already been discussed in this article: cooperating agents using threshold cryptography, and the secure computation service for execution privacy. Other solutions in which a mobile agent cooperates with a server are discussed here.

Server-Supported Signatures, S^3, are presented by Asokan et al. [1997]. They are based on hash functions and traditional digital signatures. Users only verify signatures; signature generation is done by third parties, called signature servers. The scheme is based on a hash chain of public keys: $key_{current} = h(key_{next})$. When requesting a signature, the originator will reveal the next value in the chain. As already indicated in Section 4.2, this scheme, which is the reverse of the forward-secure signature key update scheme, seems very useful for mobile agents. They can carry only a few values of the chain. If a value is compromised and misused then the signature server will detect this the next time the user or the agent requests the generation of a signature. The article discusses how to hand over authority from one signature server to another, so that roaming users (or mobile agents) can always use the nearest signature server.

Note that Server-Aided Secret Computation is a general technique in which a server aids a specific computation. Server-aided RSA protocols [Merkle and Werchner 1998] are an example. These protocols are intended for a system in which one entity is trusted (and thus has the secret) but less powerful, for example, a smartcard, and the other entity is untrusted but powerful. Whenever we have the situation where a mobile agent runs on a trusted but less powerful (part of a) platform, these protocols could be deployed.

5. DISCUSSION

This article has presented a whole spectrum of solutions that could enable mobile agents to do secure electronic transactions on untrusted hosts. These solutions were positioned in the overall framework of mobile agent security issues. We can evaluate the solutions according to various criteria. Sections 3 and 4 distinguished between general and transaction-specific solutions. To a large extent, this corresponds to a distinction between theoretical but promising versus practical and pragmatic techniques (e.g., computing with encrypted functions versus proxy certificates). Some solutions require a high level of trust, in particular, entities (e.g., signature servers) or tamper-resistant hardware, and others do not make any trust assumption (e.g., code obfuscation) or rely on a distribution of trust (e.g., threshold techniques). Some solutions protect against malicious actions (e.g., undetachable signatures), whereas others merely provide a detection mechanism (e.g., solutions for execution integrity).

Many open issues can certainly be identified for future research. Some interesting techniques should be investigated more thoroughly. For example, code obfuscation intends to protect a mobile agent for a limited amount of time, but how much time is this? There is little knowledge concerning ways of formally measuring the strength of obfuscation, similar to the foundations with respect to the various inversion problems in cryptography. This article has included many specific solutions. Not only should these be studied in more depth, but there are probably alternative solutions that have not yet been explored. Mechanisms such as threshold schemes requiring multiple agents and/or semitrusted servers have good security properties. It is, however, not clear how to deploy these schemes in today's electronic commerce scenarios. Real implementations of the solutions described in this article are therefore needed, as not that many currently exist.

The last observation leads us to the following important question: *Why* should mobile agents secure electronic transactions on untrusted hosts? While this article addressed the question of *how* mobile agents could perform secure electronic transactions on untrusted hosts, we should indeed question why they should do so in the first place. As mentioned in the introduction, Chess et al. [1997] state that mobile agents are generally a good idea. Also Kotz and Gray [1999] and Lange and Oshima [1999] share this opinion. However, as information technology is evolving very quickly, some of the arguments in favor of mobile agents probably need to be revisited. For example, wireless continuous high-bandwidth connections are currently also becoming possible for mobile devices (yet still more expensive than wired connections).

The specific case of secure electronic transactions on untrusted hosts falls into the category of "most difficult to solve" mobile agent security issues. Many solutions presented in this article therefore result in significant overheads. Further research is thus needed to indicate whether the advantages of electronic transactions by mobile agents are worth the overheads involved in securing them. This will certainly depend on the particular application and scenario, and on the security mechanisms used. In many applications a secure electronic transaction will be needed only in the final stage of a process. Although the overall process might benefit from the mobile agent paradigm, the final secure electronic transaction step might be better handled by the user in the regular client/server way. However, there might be some situations in which secure mobile agent-based electronic transactions are required, and are worth the extra costs. For example, mobile agent-based secure electronic transactions seem useful in complex and real-time scenarios where payment can be requested at any stage of the process. It might also be interesting to deploy mobile agents in the case where secure electronic transactions are desired with additional features such as anonymity. As a mobile agent can travel to other hosts, it could conduct electronic transactions on behalf of its user, while hiding the location of its user within the network.

6. CONCLUSION

The purpose of this article was to investigate if and how mobile agents can perform secure electronic transactions on untrusted hosts. A survey was presented of the problem of malicious hosts and mobile agent-based secure electronic transactions. We feel that we have covered the most representative set of existing solutions for this problem. In order to be able to keep an overview of the complete picture, our analysis was always on a conceptual level, and we never investigated deeply any particular solution. The main contribution of this article is that we have looked at the problem of secure electronic transactions for mobile agents from a very general perspective, brought many ideas together, and investigated their relevance to our problem.

Both general solutions for malicious hosts and specific solutions for electronic transactions have been discussed. These solutions should be seen as complementary. The best and most practical solution that can be constructed will be a combination of one (or more) specific schemes for secure electronic transactions, and one (or more) general solutions for malicious hosts. A combined solution is also necessary as not only the electronic transaction, but its entire context, should be protected. Note that many solutions do not offer perfect security. Some (very practical) solutions actually do not protect the secret information, but rather limit its use, and as such control the risk of compromise.

The article shows that there is currently a wide range of solutions with which mobile agents could do secure electronic transactions on untrusted hosts. As payment is a very fundamental interaction between peers, we believe that it makes sense to deploy these solutions in mobile agent-based applications, especially if payment can be required at any stage of the process, and is not just something that happens at the end (in which case it might better be handled in the regular client/server way). However, the potential benefits of mobile agent-based electronic transactions must be carefully compared to the significant overheads induced by particular solutions. In any case, how mobile agents can efficiently do secure electronic transactions on untrusted hosts remains a very challenging research question.

ACKNOWLEDGMENTS

The authors want to thank Benjamin C. Pierce and the anonymous reviewers for their valuable comments and suggestions. The authors also wish to express their gratitude to Keith Martin who provided very helpful native English support.

REFERENCES

Algesheimer, J., Cachin, C., Camenisch, J., and Karjoth, G. 2001. Cryptographic security for mobile code. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P 2001)*, 2–11.

Asokan, N., Tsudik, G., and Waidner, M. 1997. Server-supported signatures. *J. Comput. Sec. 5*, 1, 91–108.

Aucsmith, D. 1996. Tamper resistant software: An implementation. In *Proceedings of the Information Hiding Workshop '96*, R. Anderson, Ed., Springer-Verlag, New York, 317–333.

Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S., and Yang, K. 2001. On the (im)possibility of obfuscating programs. In *Advances in Cryptology—CRYPTO 2001*, J. Kilian, Ed., Lecture Notes in Computer Science, vol. 2139, Springer-Verlag, New York, 1–18.

Bellare, M. and Miner, S. K. 1999. A forward-secure digital signature scheme. In *Advances in Cryptology—CRYPTO'99*, M. Wiener, Ed., Lecture Notes in Computer Science, vol. 1666, Springer-Verlag, New York, 431–448.

Bellovin, S. M. 1989. Security problems in the TCP/IP protocol suite. *Comput. Commun. Rev. 19*, 2 (April), 32–48.

Berkovits, S., Guttman, J. D., and Swarup, V. 1998. Authentication for mobile agents. In *Mobile Agents and Security*, G. Vigna, Ed. Lecture Notes in Computer Science, vol. 1419, Springer-Verlag, New York, 114–136.

Biehl, I., Meyer, B., and Wetzel, S. 1998. Ensuring the integrity of agent-based computations by short proofs. In *Proceedings of the Second International Workshop on Mobile Agents*, K. Rothermel and F. Hohl, Eds., Lecture Notes in Computer Science, vol. 1477, Springer-Verlag, New York, 183–194.

Binder, W. 1999. J-Seal2—A secure high-performance mobile agent system. In *Proceedings of the Workshop on Agents in Electronic Commerce*, Y. Ye and J. Liu, Eds., 141–150.

Borselius, N., Mitchell, C. J., and Wilson, A. 2001a. On mobile agent based transactions in moderately hostile environments. In *Advances in Network and Distributed Systems Security—Proceedings of IFIP I-NetSec'01*, B. De Decker, F. Piessens, J. Smits, and E. Van Herreweghen, Eds., Kluwer Academic, Hingham, MA, 173–186.

Borselius, N., Mitchell, C. J., and Wilson, A. 2001b. Undetachable threshold signatures. In *Proceedings of the Eighth IMA International Conference on Cryptography and Coding*, B. Honary, Ed., Lecture Notes in Computer Science, vol. 2260, Springer-Verlag, New York, 239–244.

Cachin, C., Camenisch, J., Kilian, J., and Müller, J. 2000. One-round secure computation and secure autonomous mobile agents. In *Proceedings of the 27th International Colloquium on Automata, Languages and Programming (ICALP)*, U. Montanari, J. D. P. Rolim, and E. Welzl, Eds., Lecture Notes in Computer Science, vol. 1853. Springer-Verlag, New York, 512–523.

Chess, D. M. 1998. Security issues in mobile code systems. In *Mobile Agents and Security*, G. Vigna, Ed., Lecture Notes in Computer Science, vol. 1419, Springer-Verlag, New York, 1–14.

Chess, D. M., Grosof, B., Harrison, C. G., Levine, D., Parris, C., and Tsudik, G. 1995. Itinerant agents for mobile computing. IBM Res. Rep. RC 20010.

Chess, D. M., Harrison, C. G., and Kershenbaum, A. 1997. Mobile agents: Are they a good idea? In *Proceedings of the Second International Workshop on Mobile Object Systems: Towards the Programmable Internet*, J. Vitek and C. Tschudin, Eds., Lecture Notes in Computer Science, vol. 1222, Springer-Verlag, New York, 25–45.

Claessens, J., Preneel, B., and Vandewalle, J. 2001. Secure communication for secure agent-based electronic commerce. In *E-Commerce Agents: Marketplace Solutions, Security Issues, and Supply and Demand*, J. Liu and Y. Ye, Eds., Lecture Notes in Computer Science, vol. 2033, Springer-Verlag, New York, 180–190.

Cramer, R. 1999. Introduction to secure computation. In *Lectures on Data Security—Modern Cryptology in Theory and Practice*, I. Damgård, Ed., Lecture Notes in Computer Science, vol. 1561, Springer-Verlag, New York, 16–62.

Das, A. and Gongxuan, Y. 2001. A secure payment protocol using mobile agents in an untrusted host environment. In *Electronic Commerce Technologies—Proceedings of the Second International Symposium, ISEC 2001*, W. Kou, Y. Yesha, and C. J. Tan, Eds., Lecture Notes in Computer Science, vol. 2040, Springer-Verlag, New York, 33–41.

de Carvalho Ferreira, L. and Dahab, R. 2001. Blinded-key signatures: Securing private keys embedded in mobile agents. Tech. Rep., Institute of Computing, University of Campinas, Brazil.

De Decker, B., Piessens, F., Van Hoeymissen, E., and Neven, G. 2000. Semi-trusted hosts and mobile agents: Enabling secure distributed computations. In *Proceedings of the Second International*

Workshop on Mobile Agents for Telecommunication Applications, E. Horlait, Ed., Lecture Notes in Computer Science, vol. 1931, Springer-Verlag, New York, 219–232.

Dierks, T. and Allen, C. 1999. The TLS Protocol Version 1.0. IETF Request for Comments, RFC 2246.

Doraswamy, N. and Harkins, D. 1999. *IPSec: The New Security Standard for the Internet, Intranets, and Virtual Private Networks*. Prentice-Hall, Englewood Cliffs, NJ.

Eastlake, D., Reagle, J., and Solo, D. 2002. XML-Signature syntax and processing. W3C Recommendation.

Farmer, W. M., Guttman, J. D., and Swarup, V. 1996b. Security for mobile agents: Authentication and state appraisal. In *Proceedings of the Fourth European Symposium on Research in Computer Security (ESORICS)*, E. Bertino, H. Kurth, G. Martella, and E. Montolivo, Eds., Lecture Notes in Computer Science, vol. 1146, Springer-Verlag, New York, 118–130.

Farmer, W. M., Guttman, J. D., and Swarup, V. 1996a. Security for mobile agents: Issues and requirements. In *Proceedings of the Nineteenth National Information Systems Security Conference*.

Goldreich, O., Pfitzmann, B., and Rivest, R. L. 1998. Self-delegation with controlled propagation—or—What if you lose your laptop. In *Advances in Cryptology—CRYPTO'98*, H. Krawczyk, Ed., Lecture Notes in Computer Science, vol. 1462, Springer-Verlag, New York, 153–168.

Gong, L. and Schemers, R. 1998. Signing, sealing, and guarding Java^{TM} objects. In *Mobile Agents and Security*, G. Vigna, Ed., Lecture Notes in Computer Science, vol. 1419, Springer-Verlag, New York, 206–216.

Gray, R. S., Kotz, D., Cybenko, G., and Rus, D. 1998. D'Agents: Security in a multiple-language, mobile-agent system. In *Mobile Agents and Security*, G. Vigna, Ed., Lecture Notes in Computer Science, vol. 1419, Springer-Verlag, New York, 154–187.

Hassler, V. 2000. Mobile agent security. In *Security Fundamentals for E-Commerce*, Computer Security Series. Artech House, Chapter 20, 331–351.

Hohl, F. 1998a. A model of attacks of malicious hosts against mobile agents. In *Proceedings of the fourth ECOOP Workshop on Mobile Oject Systems: Secure Internet Mobile Computation*.

Hohl, F. 1998b. Time limited blackbox security: Protecting mobile agents from malicious hosts. In *Mobile Agents and Security*, G. Vigna, Ed., Lecture Notes in Computer Science, vol. 1419, Springer-Verlag, New York, 92–113.

Hohl, F. 2000. A framework to protect mobile agents by using reference states. In *Proceedings of the Twentieth International Conference on Distributed Computing Systems*.

Jakobsson, M. and Juels, A. 1998. X-Cash: Executable digital cash. In *Proceedings of Financial Cryptography '98*, R. Hirschfeld, Ed., Lecture Notes in Computer Science, vol. 1465, Springer-Verlag, New York, 16–27.

Jansen, W. 2000. Countermeasures for mobile agent security. *Comput. Commun. 23*, 17 (Nov.), 1667–1676.

Jansen, W. and Karygiannis, T. 1999. Mobile agent security. NIST Special Publication 800-19.

Karjoth, G., Asokan, N., and Gülcü, C. 1998. Protecting the computation results of free-roaming agents. In *Proceedings of the Second International Workshop on Mobile Agents*, K. Rothermel and F. Hohl, Eds., Lecture Notes in Computer Science, vol. 1477, Springer-Verlag, New York, 195–207.

Karjoth, G., Lange, D. B., and Oshima, M. 1998. A security model for aglets. In *Mobile Agents and Security*, G. Vigna, Ed., Lecture Notes in Computer Science, vol. 1419, Springer-Verlag, New York, 188–205.

Kim, H., Baek, J., Lee, B., and Kim, K. 2001. Secret computation with secrets for mobile agent using one-time proxy signature. In *Proceedings of the 2001 Symposium on Cryptography and Information Security*, 845–850.

Kocher, P., Jaffe, J., and Jun, B. 1999. Differential power analysis. In *Advances in Cryptology—CRYPTO'99*, M. Wiener, Ed., Lecture Notes in Computer Science, vol. 1666, Springer-Verlag, New York, 388–397.

Kotz, D. and Gray, R. S. 1999. Mobile agents and the future of the Internet. *ACM SIGOPS Oper. Syst. Rev. 33*, 3 (July), 7–13.

Kotzanikolaou, P., Burmester, M., and Chrissikopoulos, V. 2000. Secure transactions with mobile agents in hostile environments. In *Proceedings of the fifth Australasian Conference on Information*

Security and Privacy, E. Dawson, A. Clark, and C. Boyd, Eds., Lecture Notes in Computer Science, vol. 1841, Springer-Verlag, New York, 289–297.

KOTZANIKOLAOU, P., KATSIRELOS, G., AND CHRISSIKOPOULOS, V. 1999. Mobile agents for secure electronic transactions. In *Recent Advances in Signal Processing and Communications*, N. Mastorakis, Ed., World Scientific, River Edge, NJ, 363–368.

KRAWCZYK, H. 2000. Simple forward-secure signatures from any signature scheme. In *Proceedings of the Seventh ACM Conference on Computer and Communications Security*, 108–115.

LANGE, D. B. AND OSHIMA, M. 1999. Seven good reasons for mobile agents. *Commun. ACM 42*, 3 (March), 88–89.

LEE, B., KIM, H., AND KIM, K. 2001. Secure mobile agent using strong non-designated proxy signature. In *Proceedings of the Sixth Australasian Conference on Information Security and Privacy (ACISP 2001)*, V. Varadharajan and Y. Mu, Eds., Lecture Notes in Computer Science, vol. 2119, Springer-Verlag, New York, 474–486.

LOUREIRO, S. 2001. Mobile code protection. PhD thesis, ENST Paris.

LOUREIRO, S. AND MOLVA, R. 1999. Function hiding based on error correcting codes. In *Proceedings of the CryptTEC'99 International Workshop on Cryptographic Techniques and Electronic Commerce* (Hong Kong), M. Blum and C. Lee, Eds., 92–98.

LOUREIRO, S. AND MOLVA, R. 2000. Mobile code protection with smartcards. In *Proceedings of the Sixth ECOOP Workshop on Mobile Object Systems: Operating System Support, Security and Programming Languages*.

LOUREIRO, S., MOLVA, R., AND PANNETRAT, A. 1999. Secure data collection with updates. In *Proceedings of the Workshop on Agents in Electronic Commerce*, Y. Ye and J. Liu, Eds., 121–130.

MEADOWS, C. 1997. Detecting attacks on mobile agents. In *Proceedings of the DARPA Foundations for Secure Mobile Code Workshop*.

MERKLE, J. AND WERCHNER, R. 1998. On the security of server-aided RSA protocols. In *Proceedings of the First International Workshop on Practice and Theory in Public Key Cryptography*, H. Imai and Y. Zheng, Eds., Lecture Notes in Computer Science, vol. 1431, Springer-Verlag, New York, 99–116.

MINSKY, Y., VAN RENESSE, R., SCHNEIDER, F. B., AND STOLLER, S. D. 1996. Cryptographic support for fault-tolerant distributed computing. In *Proceedings of the Seventh ACM SIGOPS European Workshop*, 109–114.

NECULA, G. C. AND LEE, P. 1998. Safe, untrusted agents using proof-carrying code. In *Mobile Agents and Security*, G. Vigna, Ed., Lecture Notes in Computer Science, vol. 1419, Springer-Verlag, New York, 61–91.

NEUMAN, B. C. 1993. Proxy-based authorization and accounting for distributed systems. In *Proceedings of the Thirteenth International Conference on Distributed Computing Systems*, 283–291.

NEVEN, G., PIESSENS, F., AND DE DECKER, B. 2000. On the practical feasibility of secure distributed computing: A case study. In *Information Security for Global Information Infrastructures—Proceedings of IFIP SEC 2000*, S. Qing and J. Eloff, Eds., Kluwer Academic, Hingham, MA, 361–370.

NG, S.-K. AND CHEUNG, K.-W. 1999. Intention spreading: An extensible theme to protect mobile agents from read attack hoisted by malicious hosts. In *Intelligent Agent Technology: Systems, Methodologies, and Tools—Proceedings of the first Asia-Pacific Conference on Intelligent Agent Technology (IAT '99)*, J. Liu and N. Zhong, Eds., World Scientific, River Edge, NJ, 406–415.

O'MAHONY, D., PEIRCE, M., AND TEWARI, H. 2001. *Electronic Payment Systems for E-Commerce*, 2nd ed. Artech House.

OUSTERHOUT, J. K., LEVY, J. Y., AND WELCH, B. B. 1998. The safe-Tcl security model. In *Mobile Agents and Security*, G. Vigna, Ed., Lecture Notes in Computer Science, vol. 1419, Springer-Verlag, New York, 217–234.

RASMUSSON, L. AND JANSSON, S. 1996. Simulated social control for secure Internet commerce. In *Proceedings of the 1996 ACM Workshop on New Security Paradigms*. 18–25.

RIORDAN, J. AND SCHNEIER, B. 1998. Environmental key generation towards clueless agents. In *Mobile Agents and Security*, G. Vigna, Ed., Lecture Notes in Computer Science, vol. 1419, Springer-Verlag, New York, 15–24.

Romão, A. and da Silva, M. M. 1999. Proxy certificates: A mechanism for delegating digital signature power to mobile agents. In *Proceedings of the Workshop on Agents in Electronic Commerce*, Y. Ye and J. Liu, Eds., 131–140.

Roth, V. 2001. On the robustness of some cryptographic protocols for mobile agent protection. In *Proceedings of the fifth International Conference on Mobile Agents*, G. P. Picco, Ed., Lecture Notes in Computer Science, vol. 2240, Springer-Verlag, New York, 1–14.

Sander, T. and Tschudin, C. F. 1998a. On software protection via function hiding. In *Proceedings of the Second International Workshop on Information Hiding*, D. Aucsmith, Ed., Lecture Notes in Computer Science, vol. 1525. Springer-Verlag, New York, 111–123.

Sander, T. and Tschudin, C. F. 1998b. Protecting mobile agents against malicious hosts. In *Mobile Agents and Security*, G. Vigna, Ed., Lecture Notes in Computer Science, vol. 1419, Springer-Verlag, New York, 44–60.

Sander, T. and Tschudin, C. F. 1998c. Towards mobile cryptography. In *Proceedings of the 1998 IEEE Symposium on Security and Privacy*, 215–224.

SET Secure Electronic Transaction LLC. SET Secure Electronic Transaction Specification. Available at http://www.setco.org/.

Shamir, A. and van Someren, N. 1999. Playing "hide and seek" with stored keys. In *Proceedings of Financial Cryptography '99*, M. Franklin, Ed., Lecture Notes in Computer Science, vol. 1648, Springer-Verlag, New York, 118–124.

Shoup, V. 2000. Practical threshold signatures. In *Advances in Cryptology—EUROCRYPT 2000*, B. Preneel, Ed., Lecture Notes in Computer Science, vol. 1807, Springer-Verlag, New York, 207–220.

Spafford, E. H. 1988. The Internet worm program: An analysis. Purdue Tech. Rep. CSD-TR-823.

TCPA. Trusted Computing Platform Alliance. Available at http://www.trustedpc.org/.

Tschudin, C. F. 1999. Mobile Agent Security. In *Intelligent Information Agents: Agent-Based Information Discovery and Management on the Internet*, M. Klusch, Ed., Springer-Verlag, New York, Chapter 18, 431–446.

Vigna, G. 1997. Protecting mobile agents through tracing. In *Proceedings of the Third ECOOP Workshop on Mobile Object Systems: Operating System Support for Mobile Object Systems*.

Vigna, G. 1998. Cryptographic traces for mobile agents. In *Mobile Agents and Security*, G. Vigna, Ed., Lecture Notes in Computer Science, vol. 1419, Springer-Verlag, New York, 137–153.

Volpano, D. and Smith, G. 1998. Language issues in mobile program security. In *Mobile Agents and Security*, G. Vigna, Ed., Lecture Notes in Computer Science, vol. 1419, Springer-Verlag, New York, 25–43.

Wilhelm, U. G., Staamann, S., and Buttyán, L. 1998. On the problem of trust in mobile agent systems. In *Proceedings of the 1998 Network and Distributed System Security (NDSS'98) Symposium*.

Yee, B. S. 1999. A sanctuary for mobile agents. In *Secure Internet Programming: Security Issues for Mobile and Distributed Objects*, J. Vitek and C. Jensen, Eds., Lecture Notes in Computer Science, vol. 1603, Springer-Verlag, New York, 261–274.

Yi, X., Siew, C. K., and Syed, M. R. 2000. Digital signature with one-time pair of keys. *Electron. Lett. 36*, 2 (Jan.), 130–131.

Young, A. and Yung, M. 1997. Sliding encryption: A cryptographic tool for mobile agents. In *Fast Software Encryption—FSE'97*, E. Biham, Ed., Lecture Notes in Computer Science, vol. 1267, Springer-Verlag, New York, 230–241.

Received August 2001; revised May 2002; accepted August 2002