Mobile Agent Security – Issues and Directions

Kristian Schelderup[1], Jon Ølnes[1]

[1] Norwegian Computing Centre (NR), P.O.Box 114 Blindern, N-0314 Oslo, Norway
Jon.Olnes@nr.no

Abstract. It is obvious that a prerequisite for use of mobile agent systems in many settings is that security is taken care of. A proper trust model is necessary in order to build security. A lot of security issues arise if the trust model implies that not all hosts are trusted. This paper discusses trust models, examines the security issues, and points at possible directions for solutions in terms of security services, mechanisms, and protocols.

1. Introduction

Friendship is constant in all other things
Save in the office and affairs of love:
Therefore all hearts in love use their own tongues;
Let every eye negotiate for itself
And trust no agent.
(William Shakespeare, Ibid in Much Ado about Nothing)

Mobile code is a fairly new development in the computer science and telecommunications areas. Agents (mobile and intelligent – the intelligence issues are not discussed here) constitute an active research field, see for example [2].

Most new technologies have deficiencies with respect to security, because they are designed for functionality, with security added as an afterthought. This is not necessarily the case with mobile agents, since it is evident that any (commercially) useful mobile agent system must address security properly. This said, it is also evident, when looking more closely at the security requirements and suggested solutions, that security of mobile agents is still an area for further research. This paper analyses the security issues related to mobile agents, and points at directions for work to cope with the issues. A more detailed discussion (encrypted except for Scandinavian readers) can be found in [16].

Most agent systems refer to (up to) four elements for their security:

- a runtime environment (usually the Java Virtual Machine) for host protection;
- code signing to prove that the agent has not been tampered with;
- host authentication to prove that the agent is about to move to the intended host,
- establishment of a secure (cryptographic protection) channel over which the agent can migrate.

These are mainly geared at protecting hosts from agents (code signing and host authentication provides some protection against rogue hosts as well). If the hosts are

trusted, the four measures are sufficient, however this is not necessarily a correct assumption.
Note that the topic of code signing is often poorly understood, and security is often designed without an explicit trust model in mind (which components of the system are trusted, and for which purpose). This is discussed further below.

2. Agent platforms

An agent platform constitutes the environment where the agents exist. The primary task is to run the agents that are present in the system, and to provide well-known services and environments that the agents can utilise. The platform thus realises a virtual machine for agents, with a set of operations available. The virtual machine may of course be a particular operating system, meaning that agents can only run on this system, but preferably agents should be operating system independent.
The design and realisation of agent platforms differs a lot depending on their technological foundation. An agent system can be realised in a fully object oriented way, where each agent is represented by an agent object, and the agent platform likewise consists of objects with well-known or published interfaces. This approach is taken by OMG's MASIF (Mobile Agent System Interoperability Facility) proposal [14] which builds on the CORBA platform.
A platform may also be quite simple, like TACOMA [9] where agents run directly on the host's operating system, but with well-known interfaces available.
However, most agent systems today base the agent platform on the Java Virtual Machine (JVM). From a security viewpoint this is a sound design, as JVM can provide strong protection of the host system [5], dependent on the security policy configured in the JVM installation. Also, the agent paradigm fits well with the Java idea of mobile code (write once, run anywhere).

3. Agents and cryptography

Proper authentication, integrity, and confidentiality protection in open networks demand use of cryptography. Three of the usual security measures in agent systems (see introduction) are based on cryptography, and aim at protection of (trusted) hosts. If hosts are regarded as suspicious, which is actually a sound standpoint in many cases, the agents themselves may need protection by cryptographic means.
Note that an agent can never carry private or secret (symmetric encryption) keys since they will be open to the hosts (even if hosts are trusted, this is stretching the trust too much). An agent can carry public keys, provided that they are protected against tampering by certificates. Today's agent systems use the hosts' cryptographic facilities through defined interfaces. This means that the host will control all cryptographic processing, including verification of certificates and other signed material, as well as authentication procedures that are based on cryptography. The agents must trust the host to perform these operations correctly.
An agent may carry its own cryptographic routines (signed and trusted code), but this does not help, since the host controls the execution, and may run one of its own routines instead of the one that followed the agent.

Ultimately, if hosts are not sufficiently trusted, the only possible protection measure is to embed the agent's execution environment in trusted hardware, where a certificate from a trusted entity proves that the hardware is genuine. The environment would encompass cryptographic routines. This is discussed later in this paper under the hat of integrity protection.

4. A cautionary note on code signing

As stated, code signing is often poorly understood, although frequently referred to as an important protection measure. Firstly, only static parts of the agent can be signed. The program code can be assumed to be static. If modifications to the code were allowed, it would anyway be virtually impossible to verify that the agent keeps its integrity properties. However, not much apart from the code will in general be static. As one example, the list of hosts that the agent shall visit can well be altered, by adding or removing hosts based on information that the agent obtains on its way. Code signing detects tampering of static information, and in this aspect protects both the hosts and the agent itself. However, as discussed later in this paper, many aspects of agent integrity and confidentiality are still left.

The usual flaw with code signing schemes is that authentication, trust, and authorisation / roles are not separated. Verification of the signature under a trusted certificate is proof of identity. It is not necessarily proof of trustworthiness, nor authorisation for a certain role or certain access rights.

Signatures produced by several roles may be useful for agents. There is a need for a standardised way of representing these in certificates. Roles may be:

- the agent's authority;
- the author of the agent code (person or software house);
- a vendor that sells the agent;
- a third party that has performed a security evaluation of the agent.

5. Trust modeling for agent systems

Virtually all agent systems regard an agent as potentially suspicious, and virtually all systems build (deliberately or not) on a high level of trust in the hosts. In many cases, for example inside an organisation, this trust may be well founded.

A more cautious model is to regard a host as trusted with respect to its own information (the host will anyway supply information totally at its own discretion), but not with respect to information supplied by other hosts or carried by the agent. As one example, a host that offers some service to an agent that collects information from a number of competitors, might well destroy or modify information from its competitors, or alter the agent's list of host or otherwise arrange to skip competitors. On the other hand, the host is as interested as the agent in preserving the integrity and confidentiality of the information it supplies itself.

An even more paranoid model can take into account that a host may behave unexpectedly, e.g. because it is the victim of an intruder. In this case, a lot of new threats emerge, the most important ones probably related to the agent as a convenient vehicle for leaking information unnoticed. Furthermore, one can easily envisage attacks on agents where different hosts co-operate. When a protected and authenticated channel is used to protect an agent on the move from one host to another, the authentication and key negotiation is actually done by the hosts. Thus, it is fairly easy for a host to ship the agent off to another host than the intended one. Also, weak keys may be used, or keys may be deliberately leaked to eavesdroppers.

The same goes when a host signs or encrypts information that is carried on with the agent. This protection may be done in a way that makes it possible for a particular other host to read, delete or alter the information.

Protection methods against rogue hosts may rely on defining some hosts as trusted, and use these as intermediaries when travelling between untrusted hosts.

Another, usually neglected, issue is the level of trust that the agent’s authority has in the agent. An agent that is written by the authority itself, or has a signature from a trusted evaluator, will be more trusted than “a piece of code obtained from some source”. This can be reflected in the authorisations that the agent can be given.

6. Authorities

An agent runs on behalf of some authority, which ultimately will always be a person. An authentication of this person, at some time and place, is necessary. However, the identity of the person need not be revealed to the hosts that the agent visits or to other agents. The agent may represent a role (which can be taken by a number of people) or an organisation, or it may even be anonymous, but carry proofs of its authorisations. The role or authorisations will be used for access control, while the real, authenticated identity of the authority is needed for the traceability of the system, since an agent’s authority should be held responsible for the agent’s actions. The authority will also be the agent’s primary source of authorisations, i.e. rules governing what the agent is allowed or not allowed to do.

7. Security requirements

Risk analysis

A risk analysis consists of an analysis of threats, their implied risk, which can be viewed as the probability of being hit, and estimated impacts of such realised threats. Security measures, which imply cost, must be weighted against this analysis. A real risk analysis can only be performed for real systems, used for particular purposes, were both the environment and the value of the assets are known. An analysis for a “generic” technology, like agent systems in general, can only reveal possible threats and risks. In real implementations for specific purposes, threats may be too unrealistic, or their effects may be negligible, varying from use case to use case.

Threats may have different sources. In this paper, we only consider intentional attacks on systems or information. However it is clear that unintentional malfunctioning (bugs) often can cause the same effects.

Security threats relate to attacks on the integrity, confidentiality, or availability of systems or information. Additionally, there is a need to be able to trace events in the system back to the responsible parties with some degree of certainty.
Threats are towards the hosts that an agent visits, but it is also necessary to consider threats to an agent from malicious hosts or from other agents.

Threats to a host

It is obvious that an agent may be used to attack hosts. The threats can be against:

- data integrity or confidentiality – an agent must be prohibited from access to any information or operations on information that it is not authorised for;
- system integrity – an agent shall not be able to cause unauthorised changes in the behaviour of the host;
- availability – an agent shall not be able to cripple the performance of the host.

The main security measure to ensure integrity and confidentiality is proper access control. Other measures may enhance this, like checksums and encrypted storage.
It is well understood that today's off the shelf operating systems cannot withstand attacks from malicious code that has been successfully loaded on the system. Such code will circumvent the normal access control procedures one way or another. This is one of the main reasons for agent systems to be based on the Java platform, using the Java virtual machine to limit agents' behaviour.
In most cases, the hosts will have requirements for logging of the actions of an agent, and the nature of the agent like its identity, role, authorisations, and which authority it ran on behalf of. While simple log files will be sufficient in most cases, requirements for non-repudiation services may also emerge.

Threats to an agent

When an agent is loaded onto a host, this host gains full control over the agent. A malicious host may:

- change the agent code, e.g. to launch attacks on other hosts;
- alter, delete or read / copy (confidential) information that the agent has recorded from previous hosts.

If an agent interacts with another agent (usually both in the same host), the other agent may attempt similar attacks. The other agent's possibilities will be limited by the host, but it is of course possible for the host to help on the attack instead.
An agent will usually keep a trace of the hosts it visits, and relate this to information that it gathers along its route. It may also have other traceability requirements, to the extent that the hosts (or some authority) may need to sign information for non-repudiation purposes.
It is virtually impossible to prevent a host from capturing an agent and prevent it from further execution. Measures may be taken to track the progress of agents to at least detect such situations, and possibly identify the responsible host.

8. Security services and mechanisms

Threats are encountered primarily by security services, which may be realised by different security mechanisms. The security frameworks of ISO and ITU-T [8] are convenient references for security services and mechanisms. Below, the applicability of services and mechanisms with respect to mobile agents is analysed. Additional security measures for integrity, availability and traceability are also analysed.

Authentication

Authentication in an agent system shall provide proofs of the identity of:

- agents – provided that they have names and can be identified;
- hosts (and potentially services);
- authorities / roles.

It is assumed that all hosts and authorities possess a secret / public key pair, the ability to sign by use of the private key, and a public key certificate with an issuer that is trusted by all parties. Further discussions on certification, including naming of hosts, authorities and roles, are considered outside the scope of this paper.

[16] includes a short study of the possibilities of using a Kerberos-style [19] authentication scheme, relying on an on-line server. The conclusion is that this seems to be an inferior approach compared to use of public key cryptography.

Agents cannot carry private keys, and thus cannot sign. The identity of an agent can be authenticated e.g. by including the agent name in the agent itself, and having some trusted authority sign the information.

Agents will in most cases have names, but the (global) significance of the names will vary, and thus the need for authentication. In the OMG MASIF proposal [14], an agent is named by a unique identification of the agent's authority combined with a unique identification of the agent within the domain of the authority. Each instance of an agent class is assigned a unique name. This simplifies the identification of the authority, since the authority's name is a part of the agent name. The authority name in this aspect may actually be substituted by a role name.

With respect to authentication of hosts, the agent must trust the host on which it currently resides for authentication of the next host in the chain. Including authentication code in the agent does not help, since this can easily be spoofed by the host. This opens possibilities for attacks where several hosts co-operate. Tracing procedures can attempt to encounter this, e.g. by use of signed confirmations [3].

Authentication may be based on:

- knowledge (e.g. passwords);
- possession (e.g. of a smartcard) – this is not relevant for agents;
- intrinsic properties (e.g. biometrics);
- information from a trusted third party.

An agent may actually use passwords towards a host, provided that each (user id.) and password is encrypted under the host's public key. This leaves a static list of hosts, or the option to travel via a trusted host to insert new hosts and passwords. This is not considered as a particularly good solution, especially since public key cryptography is needed anyway.

An agent’s code and other static information can be said to constitute intrinsic properties of an agent instantiation. An agent’s authority (or a role) can be authenticated through a signature over (selected parts of) the intrinsic properties of the agent. This will also authenticate the agent, if the agent name is included, and one trusts the authority. Alternatively, authentication of the agent can be done through a separate signature by another trusted role (see section 4).

Agent name (authority / role id. + agent id.)	
Home system identification	
Error handling information	
(Formal) description of purpose	
Authentication certificates (authority / role and other)	
Authorisations (e.g. privilege attribute certificates)	
Ref. static part 1	Hash value
Ref. static part 2	Hash value
Ref. static part N	Hash value
Signature(s)	

Fig. 1. Example of an agent passport

A promising approach is to define agent passports [3] as shown in Fig. 1. In this paper, extensions to the passport defined by [3] are suggested. A passport should always include the agent name (authority id. plus agent id. with the MASIF approach), identification of the agent’s home system, and a list of references to static parts of the agent with the corresponding hash value of each part. The passport is signed by the agent’s authority, possibly also by other roles. A table of content (not part of the passport, since it may change) lists the various parts of the agent, including size, description of content (e.g. code), and whether they are critical to agent processing or not. Additionally, the passport can include instructions for error handling, a description (preferably formal) of the agent’s purpose, (references to) identity certificates, and authorisations.

Access control

Access control in an agent system shall protect the following access categories:
- agent to agent platform, to ensure that only authorised agents can run on a host;
- agent to host (via the agent platform), e.g. for access to information in the host;

- host to agent, which can only be controlled if the agent platform consists of trusted hardware;
- agent to agent, controlled by the agent platform.

The first observation is that the agent platform must guarantee that all access attempts are subject to the access control procedures, i.e. that access control cannot be circumvented. This is usually accomplished by use of the Java virtual machine.

The second observation is that interaction between agents must rely on the agent platform with respect to access control.

The third observation is that the host has complete control over the agent unless the agent platform consists of a co-processor implemented in trusted hardware, as discussed other places in this paper. If the host has complete control, the only measure that can be used is later detection of access violations.

The main problem related to access control in agent systems is to describe and validate an agent's authorisations in a way that makes the host sure that it is safe to let the agent perform its task[1]. In practice, it is often desirable to have agents that can perform substantial tasks on behalf of their authorities.

"On behalf of" is an important statement here. The agent has certain authorisations that it needs to perform its tasks. The main source of authorisations will be the agent's authority, but other sources are also relevant. Typically the authorities for the agent system and for the relevant security domains [11] will set basic rules that all authorisations need to obey.

The problem we need to solve is about delegation (from authority to agent). In the simplest case, the agent can run with the full privileges of its authority, using an access control list (ACL) scheme based on the authority's identity. In many cases, the agent's authorisations should be considerably narrower.

Authorisations can either be in the form of ACLs that are stored in the hosts, or represented as capabilities that are carried with the agent[2]. The agent may also carry authorisations for access to services and information that the agent can offer to other agents or to hosts (the latter makes sense only if access control for host to agent accesses is effective). This should be static information (e.g. ACLs) and included in the agent passport.

Authorisations that are carried with the agent must be cryptographically protected. This can be achieved by putting them in the agent passport, but presumably (this needs further analysis) a better solution will be use of privilege attribute certificates (PAC) as for example suggested by the SESAME project [13]. SESAME also supports delegation of authority, which should be seen as a prerequisite in an agent system setting. The use of on-line privilege servers for PAC issuing might be a viable alternative for management of authorisations in an agent system.

In any case, it should be evident that the access control procedures and interactions (and for that sake other protocols) for an agent system need formal specification and

[1] The same problem is encountered for example when a Java applet is loaded on a host, and today applets can only have very restricted access rights.

[2] Authorisations may be weighted against context based rules. Label based access control is not considered here.

verification, e.g. by use of the methods described in [1]. Preferably the specifications should compile into a consistent representation in the agent platform. The results of [12] should be very useful in this aspect, especially if the agent system is specified according to the ODP framework. The specifications may be preloaded in the agent platform, or they may be carried in the agent passport.

Integrity and confidentiality

Integrity and confidentiality of information in (and about) the host systems must be preserved by proper access control, and a virtual machine that ensures that agents cannot circumvent the access control procedures, as discussed above.

An agent may carry information that needs protection with respect to:

- external parties that are not involved in the agent's operation;
- (some of) the hosts it visits;
- other agents.

The agent information consists of:

- the agent's program code and other static information;
- the agent's state information (including list of hosts);
- data that the agent carries (gathered along its route).

Protection against external parties has two components:

- protection against eavesdropping and modification when an agent migrates from one host to another;
- protection of the agent when resident on a host.

Protection under transfer can be ensured by establishing a secure communication channel, e.g. using the SSL protocol [18]. Both hosts are authenticated during the establishment phase, and the transfer is protected by cryptographic means. Alternatively, the agent may be protected by a cryptographic message format, like PKCS#7 [10] or PGP [15], specifying the intended host as the receiver. Note the discussion in the authentication section above about the trust issues related to authentication.

If the hosts are trusted, and the hosts' protection mechanisms are sufficient to protect against external attackers, an agent needs no particular measures to protect itself inside the host systems.

If the agent does not trust hosts with respect to its own integrity and confidentiality, there are two protection methods:

- resort to a trusted host;
- use cryptographic protection.

For each host that the agent visits, a communication channel may be established to a trusted host (which may be the agent's home system), and all information that the agent does not need later on may be transferred. Alternatively a protected message may be sent to a trusted receiver, e.g. the agent's authority. If this procedure is carried out for each host, a host that is trusted with respect to its own information may carry out the procedure on behalf of the agent. This protects only data that the agent has collected, not the agent itself (code and state), which to a certain extent can be protected by its agent passport and code signing.

Alternatively, the agent may at times visit a trusted host. A lot of the state information (like the list of hosts) may be kept in the trusted host, protected from other hosts, and collected information may be unloaded before the agent moves on. The agent can also be checked for unauthorised modifications. Ultimately, the agent may migrate via the trusted host each time it moves from one host to another, but it must be confessed that this extreme scenario defeats many of the arguments for using a mobile agent system in the first place.

With respect to cryptographic protection, three elements must be separated:

- code and other static information;
- collected data;
- state and other dynamic information.

Static information can be integrity protected by "conventional" code signing. In many cases (like program code), confidentiality of static information does not make sense. In other cases static information may be encrypted under the public keys of the entities that are allowed to access the information.

Collected data can be integrity protected by a signature from the host that supplied the data. If the data comes from another agent, some host must sign on behalf of that agent. This also provides authentication of the source, e.g. for non-repudiation purposes. Collected information can be encrypted using the public key of the agent's authority. If information shall be used during visits at intermediate hosts, it can be replicated within the agent, and encrypted separately using the public keys of each host. Unauthorised insertion of information can be detected through these measures, but unauthorised deletion will pass undetected. A suggested solution [3] can be to demand a signed confirmation when an agent moves from one host to another. A confirmation provides a snapshot of what the agent looked like at that point, and potentially this can be used to track changes.

It is claimed [4] that it is impossible to protect an agent's state information on a hostile host. The agent cannot verify correct execution of its own code, and the host can have the agent execute any code. The only solution to this [21] seems to be introduction of secure hardware (co-)processors which protect the agents during execution. The hardware needs to be properly authenticated as such, to avoid loading agents on faked equipment. The possible effects of an attack on the agent's state information with respect to other hosts need further analysis. It is quite likely that scenarios can be constructed, where the effects can be severe.

Traceability

It is frequently desirable to trace events in a system. Requirements may even be production of proofs of events, by use of non-repudiation mechanisms. It should be possible to trace events relevant to hosts and relevant to agents and their authorities.

In a host, ordinary event logs will usually be sufficient, possibly enhanced by snapshot information about the agent on arrival and departure, like the confirmations suggested by [3]. A security relevant event may be followed by a "core dump" of the agent to track the cause of the security breach. Agent passports will typically be logged to

prove the authorisations that the agent carried, and the entities that issued them. Note that the agent cannot sign information, and thus the hosts can never get signed evidence, except from passports, and signatures from other hosts.
An agent's authority may be interested in a trace of the hosts that the agent visited, the information that was supplied by the different hosts, and possibly actions that have been performed to the agent at the hosts (e.g. adding a new host in the list of hosts). As indicated above, this can be achieved by having the host sign the information that it supplies. Receipts may be used to gain snapshots of the agent's state, typically on arrival and departure. The receipts may be carried with the agent, or transferred to the agent's authority via a communication channel from the host. The value of receipts if several hosts co-operate on an attack remains to be analysed.
Ultimately, agents may always migrate via a trusted host that can check for tampering and otherwise log proofs.

9. Availability

Availability of hosts

There are two main issues with respect to an agent attacking (either due to errors or intended, malicious behaviour) the availability of a host:

- unauthorised modification (host integrity) – this has been treated earlier in this paper;
- unduly consumption of resources, notably processor, memory, disk space, and network capacity.

Resource policing should be a part of the agent virtual machine on the host, and in turn this can more or less be built on services from the operating system. In any case it is a difficult problem to solve in general, as agents may have highly varying requirements, e.g. with respect to the amount of network traffic they generate.
An agent may be authorised to a fixed amount of resources, like in AgentTCL [6]. However, AgentTCL allows an agent to send requests to a resource manager for extended resources.
Alternatively limits may be decided by (a combination of) the host, the agent's authority, and the agent itself (declaring resource needs), as first suggested by Telescript [20]. The host will have the final word, to ensure that it can live up to the resource demands. This approach seems to have been adopted by most other agent systems that address the availability question.

Availability with respect to agents

A host can halt or unduly slow down execution of an agent at will, or it can prevent agents from migrating to all or certain other hosts. An overloaded host, or a system that crashes while the agent is on board, may cause the same effect. This cannot be prevented, and the only way to detect such events is to combine timers in a trusted system – e.g. controlled by the agent's authority – with a record of the hosts that the agent has visited. Thus one may pinpoint the host where the agent is trapped.

We have not discussed persistence requirements in this paper. A possibility (likely to be quite costly) is to keep a clone of the agent in a well-known state, preferably just before or after execution on a particular host. If the agent is lost, one may rollback the actions of the agent to this state, and then restart the clone (preferably also skipping over the rogue or malfunctioning host). If the original agent is later released from its trap, measures must of course be taken to kill it. Some suggestions for agent fault-tolerance by exploiting replication can be found in [17].

Another persistence issue is whether an agent can survive host crashes, and continue operation when the system is fixed. In this case, there should definitely be timing constraints involved, to avoid network beasts like “flying dutchmen” [7].

10. Some thoughts on performance

It is clear from the discussions in this paper that a full-fledged security system for mobile agents, taking untrusted hosts into account, will carry a high performance penalty. One characteristic of mobile agents is that they are autonomous, and run without any direct user intervention. Thus, there is often no response time requirements with respect to end users, and even processing times in the range of seconds may be acceptable in most cases for loading an agent safely onto a host.

There are of course cases where performance is more critical. In such cases, if the security processing is the bottleneck, the main optimisation can be gained from ensuring that hosts can be trusted.

Another performance issue arises when we look at the system from the viewpoint of a host. If processing of each agent is too resource consuming, this may of course hamper the scalability of the system.

11. Conclusion

This paper raises several issues related to the security of agent systems, notably with respect to trust modelling and procedures for authentication, access control, integrity, and confidentiality. The use of agent passports that can include information relevant to authentication and access control, and that enhances today’s code signing procedures, is advocated. For access control, formal specification should be used to ensure a consistent solution. The important issue is delegation of authorisations, where the use of privilege attribute certificates should be examined.

For trust modelling, the main issues discussed in this paper are related to the level of trust that agents place in the hosts. Integrity (and confidentiality) of agent information is a very difficult problem, unless the hosts are trusted. Ultimately, there is a need for trusted hardware implementations of the agent platform, a requirement that may actually be met through the development of Java chips, when the agent platform is based on JVM. Several other protocol alternatives are sketched, using cryptographic protection and trusted hosts that the agent may visit underway, or transfer information to. These protocols should also be formally specified and verified. The paper includes some discussions on traceability, availability, and performance.

References

1. Abadi, M., Burrows, M., Lampson, B., Plotkin, G.: A Calculus for Access Control in Distributed Systems, ACM Transactions on Programming Languages and Systems, 15(4) (1993)
2. ACTS Agent Cluster: http://www.fokus.gmd.de/research/cc/ima/climate/entry.html
3. Chess, D., Grosof, B., Harrison, C., Levine, D., Parris, C., Tsudik, G.: Itinerant Agents for Mobile Computing, IBM Research Report RC 20010 (03/27/95) (1995)
4. Farmer, W.M., Guttman, J.D., Swarup, V.: Security for Mobile Agents: Issues and Requirements, Proceedings of the National Information Systems Security Conference (1996)
5. Fritzinger, J.S., Mueller, M.: Java Security Whitepaper, Sun Microsystems Inc. (1996)
6. Gray, R.: AgentTCL: A Flexible and Secure Mobile Agent System, Proceedings of 4th Annual USENIX TcL/Tk Workshop (1996)
7. IBM Research: Things that Go Bump in the Net, IBM Research Division, T.J. Watson Research Center (1995)
8. ITU-T X.810-816 | ISO/IEC 10181/1-7: OSI - Security Frameworks for Open Systems (1996)
9. Johansen, D., van Renesse, R., Schneider, F.B.: An Introduction to the TACOMA Distributed System, Version 1.0, Technical Report 95-23, Department of Computer Science, University of Tromsø (1995)
10. Kaliski, B.: PKCS#7: Cryptographic Message Syntax Version 1.5, RFC2315 (1998)
11. Karjoth, G., Lange, D.B., Oshima, M.: A Security Model for Aglets, IEEE Internet Communication (1997)
12. Kristoffersen, T.: A Security Architecture for Open Distributed Processing, Dr.Scient. thesis 14, University of Oslo (1998)
13. McMahon, P.V.: SESAME V2 Public Key and Authorization Extensions to Kerberos, Proceedings of the ISOC Symposium on Network and Distributed Systems Security (1995)
14. OMG Joint Submission by Crystaliz Inc, General Magic Inc, GMD FOKUS, IBM, TOG: Mobile Agent System Interoperability Facility (1997)
15. Pretty Good Privacy (PGP) international homepage: http://www.ifi.uio.no/pgp
16. Schelderup, K.: Security of Mobile Agents (in Norwegian), NR Report 921, Norwegian Computing Centre (NR) (1997)
17. Schneider, F.B.: Towards Fault-tolerant and Secure Agentry, Proceedings of the 11th International Workshop on Distributed Algorithms (1997)
18. Secure Sockets Layer (SSL): http://www.netscape.com
19. Steiner, J.G., Neumann, C., Schiller, J.I.: Kerberos: An Authentication System for Open Network Systems, Proceedings of the 1988 Winter USENIX Conference (1988)
20. Tardo, J., Valente, L.: Mobile Agent Security and Telescript, Proceedings of the IEEE Compcon ’96 Conference (1996)
21. Yee, B.S.; A Sanctuary for Mobile Agents, DARPA Workshop on Foundations for Secure Mobile Code (1996)