

# Security Enhancements for Mobile Agents Platforms

HAMED AOUADI AND PR MOHAMED BEN AHMED

RIADI, ENSI, University of Manouba

## Summary

Mobile agents technology is considered as a promised solution for distributed electronic services on the Internet. However, agents that roam the Internet can be attacked on their roads or on the visited hosts. In our researches we interest in protecting mobile agents on visited hosts. So, this paper will in first part present the state of the art of securing mobile agents against malicious hosts and in second part an approach that we suggest. In such state of the art, we identify strong and high cost approaches based on the use of additional hardware and weak and low cost approaches based on software solutions. In our approach, we upgrade mobile agents platforms by security enhancements to produce a secure execution environment that we call SVM (Secure Virtual Machine). Thus, an agent is ciphered, signed (in part) and time stamped. On visited host, the agent will be exclusively executed by the secure virtual machine SVM without decrypting it. The cipher algorithm that we use is based on the segmentation of the agent, obfuscation of the segments, a symmetric encryption of every segment and the mess-up of segment execution sequence. The security enhancements will be introduced in the loader of agents, the runtime and network access part. Finally, the entire platform will be itself protected by a smart card.

## Key words:

Mobile agents, cryptography, security, SVM.

## I. INTRODUCTION

Mobile agents are identified as the basic platform for future frameworks of distributed electronic services[hoh97]. They can be defined as programs that migrate from a host to an other to achieve a task specified by them owner[wil98]. Platforms executing agent can be malicious and try to discover the agent intention, to read

data transported by agent, to spy agent execution or modify the agent data, code or behaviour. In literature many approaches are proposed to solve the problem of agent execution on malicious hosts. In this paper we will present first the state of the art in securing mobile agents on malicious platforms then a synthesis of our approach.

## II. SECURITY CONCERNS

Mobile agents are agents that migrate from one site to another to achieve a specific task. This migration can be over an unsecured network. In such case the problem can be treated as a security network problem and solutions like secure communication protocols can be used. Migrated to a host, the agent can interact with another agent that isn't obviously trust, also hosts where agent is executed can be malicious and try to tamper whit the agent. In summary, security concerns are presented in the fig.1 and can be classified as follow[Hoh97]:

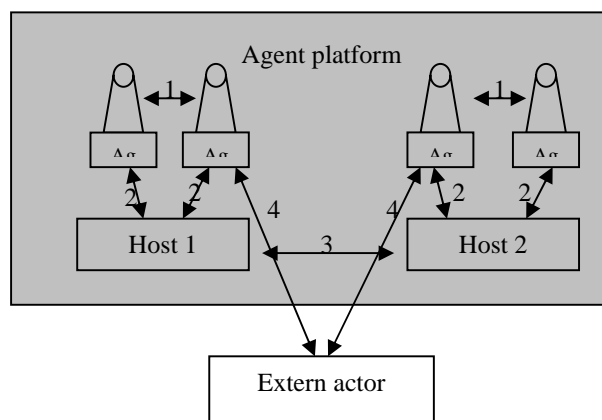


Fig1. security concerns

1. Security concerns between agents. In this class, we search to protect an agent against attacks from

other agents, we distinguish active attacks[Sah98] which modify agent code or data and passive attacks which try to discover agent secrets.

2. Security concerns between agent and agent platform. The host has total access to the agents which execute. In one hand the host can analyse the agent code, data or state and tamper with. In the other hand the agent can contain malicious code and causes damages on the host.
3. Security concerns between hosts. This class is out the scope of this paper.
4. Security concerns between agent and extern actor. Roaming the network an agent can be kept, modified or copied by a hacker. In this class we can apply a mechanism of network security like IPsec[RFC2401], SSL[Netscape], SHTTP[Res98] or other.

**III. RELATED WORKS**

In this section we briefly present approaches of securing mobile agents that we can classify in:

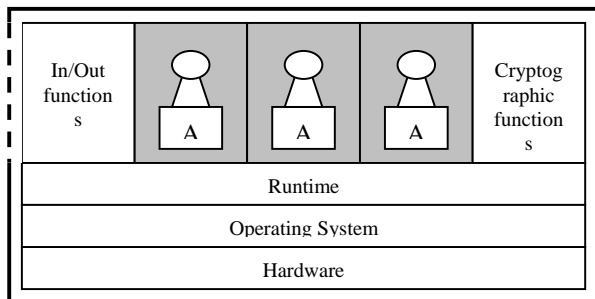
- Hardware based solutions and
- Software based solutions.

**A. Hardware solutions:**

In this case agent execution is only possible if a special peripheral exists. The agent test the existence of the peripheral on start or while executing the agent[Hoh97]. In this scheme the agent can be tampered if we delete (jump) the test of peripheral from the agent code.

**1. Trusted Processing environment:**

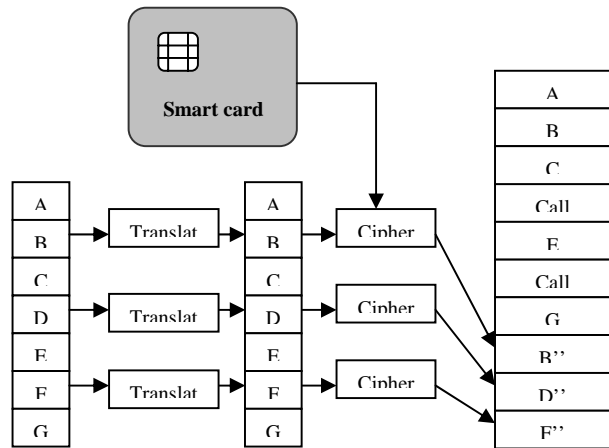
This approach is proposed by Wilhelm[wil98] and consist in the total execution of the agent inside a special peripheral called TPE (Trusted Processing Environment). The agent communicates with the visited site through a secure logical interface and migrates from one TPE to another in encrypted asymmetric form.



**Fig 2 The trust processing environment TPE**

**2. Smart cards:**

The use of smart cards to protect software is proposed by Mana [man02]. The idea consists in the segmentation of the agent, some segments will be encrypted with the public key of the card. While executing the agent the site transmits the encrypted segment to the card where it will be decrypted and executed inside it.



**Fig 3 : Smart card based security**

**B. Software solutions:**

The purpose of this class of approaches is to provide security of agents only with software mechanisms, so reduce costs and we provide maintainable products.

**1. Obfuscation:**

Obfuscation, proposed by Hohl [Hoh97] has for object the generation from an ordinary agent A another agent A' equivalent in functions but difficult to analyse. The idea consist in the violation of software engineering rules like the use of GOTO instead of recommended loops, the use of insignificant variable names, the replacement of procedure calls by the procedure body and the insertion of useless code.

Article Price Quantity

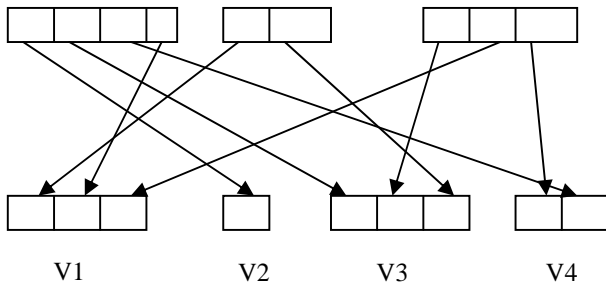


Fig 4 : Generation of insignificant name variables

**2. cryptographic function:**

In the approach proposed by Sander and Tshudin[ST98], an agent A implements a function  $f$  that will be encrypted by an algorithm E to obtain  $E(f)$ .  $E(f)$  conceals details of A and will be implemented by a program  $P(E(f))$  that is considered as a new agent B. The agent B migrates to the next site where it will be executed on a value  $x$ , thus we have  $P(E(f))(x)$  that will return to the source site that execute the decryption algorithm  $E^{-1}(P(E(f))(x))$ . For an homomorphism function  $f$  the result is  $f(x)$ .

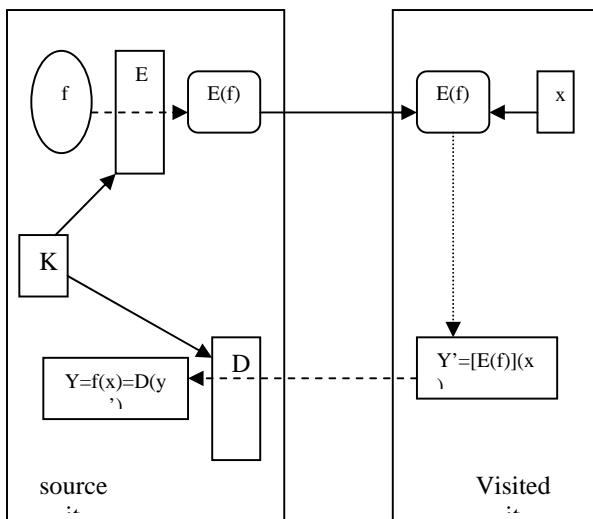


Fig 5 cryptographic function

**3. Execution traces:**

In this approach every site visited by the agent generates a trace of the agent execution [Vig98]. This trace contains every code line, every variable and every external variable read by the agent. Before the migration of the agent, they calculate the hash of the trace (with a hash function like SHA[FIPS180]), sign that hash and associate the result to the agent. A light version of this solution consists in the agent code split in white segments and black segments[Lau01]. The trace is calculated only on the black segments where the agent interacts with the visited platform.

**4. State appraisal:**

In this approach, Farmer and al.[FGS96a] suggest a mechanism allowing an agent to evaluate his privileges on a visited site. So the agent limits the actions that he can do and that he is responsible for function of the evaluated state. The state appraisal is a complex function computed from different state variables and will be executed when the agent arrives on a site.

A similar approach was suggested by Jansen[Jan00] [Jan01][Jan201] [Jan3 01] and associates an X.509 [ISO9594-8] certificate to the agent. In which certificate the creator of the agent define the actions that his agent can do and for which he is responsible. In his approach, Jansen defines the attribute certificate in which the creator of an agent defines the behaviour of his agent and policy certificate in which the site defines his security policy in front of visitor agents.

**III. COMPARISON OF APPROACHES:**

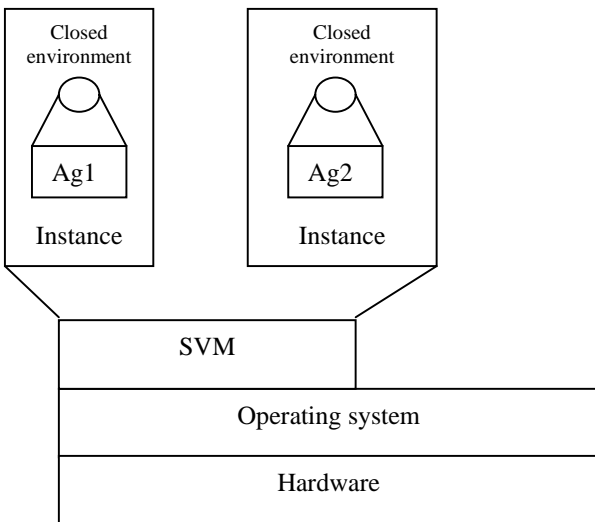
We compare the approaches presented in function of the security concerns provided and the added cost of the approach. In terms of security concerns we are interesting in the *confidentiality of execution* and the *integrity of execution* and in term of added costs we examine the *added time execution*, the *additional communications* and the *size code augmentation*.

<b>Criteria</b> <b>Approaches</b>	<b>confidentiality of execution</b>	<b>Integrity of execution</b>	<b>Size code augmentation</b>	<b>Added time execution</b>	<b>additional communications</b>	<b>comment</b>
<b>Trust Processing Environment</b>	High	High	Size code not modified	Depends on TPE	None	The performances of the site depends on the TPE that is difficult to maintain
<b>Smart card</b>	High	High	Call card instructions are inserted	Low and caused by card access	None	every site must have all cards related of all visitor agents
<b>Obfuscation</b>	High for a limited amount of time	High for a limited amount of time	Relatively acceptable if dead codes are inserted	No added time	None	The robustness of the approach is not proved
<b>Cryptographic function</b>	High	High	Relatively acceptable	Relatively acceptable	The agent must return to the source site after every visited host	The approach is limited to agents which implement polynomial functions
<b>Execution traces</b>	Not provided	verifiable	Acceptable	Acceptable	None	The approach provides only the verification of a correct execution of the agent
<b>State appraisal</b>	Not provided	Partially provided	Relatively acceptable	acceptable	None	The approach provides only the verification of the real intentions of the agent owner

**IV. Our approach:**

The approaches presented above split into strong and high cost hardware approaches and weak and low cost software based approaches. The purpose of our approach is to provide a maintainable software based solution that is as robust as hardware solutions.

In this way we upgrade the execution platform of agents by security enhancements in order to produce a software solution as robust as the Trusted Processing Environment (TPE) of Wilhelm. Thus the platform upgraded by security enhancements produce a secure software machine that we call SVM (Secure Virtual Machine). The secure virtual machine SVM will create a protected execution environment for every agent (Fig 6) witch will provide the same protection services of the TPE of Wilhelm. An agent will be written in an ordinary language then transformed by a software called agent transformer to a C-agent equivalent in function but cryptographically protected.



**Fig 6: The secure Virtual Machine SVM**

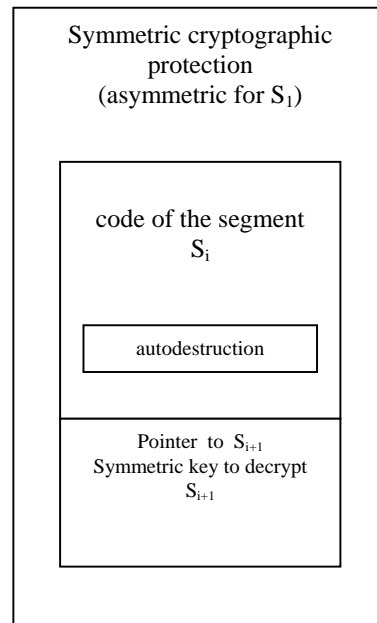
**The virtual machine SVM:**

The virtual machine SVM is a software layer installed between the operating system and the agent environments. This security layer mill be obtained by patching the component of agent platform with security mechanisms. In this way we will install a function that verifies the certificate and the source of a coming agent in the loader. The runtime will be modified in order to execute C-agents instead of ordinary agents and the network access component will be upgraded with the

verification of next site certificate, the signed stamps manager and transport layer security mechanism. This security layer (SVM) will be installed on potentially visited sites. The sites to be visited by the agent must have a certified SVM. On a site, SVM receive an agent and creates an instance of SVM to execute only this agent in an allocated memory space called closed environment. In his closed environment, the agent interacts with the visited site only via a secure interface. So, the communication manager must be enhanced with secure communication mechanisms (authentication, notification, signature). The agent also will contain many instructions randomly distributed in his code and entail the agent migration and the destruction of the closed environment. These instructions are invoked at the end of the agent execution, after a tamper detection, or if the agent time validity expires. Before migration the agent will be associated with a signed stamp that contains the current site time and the next site time.

**The agent transformer:**

The object of the agent transformer is to generate from an ordinary agent Ag a cryptographic agent C-Ag. The transformation consists in the segmentation of Ag in a set of segments  $S_i$ , every segment will be obfuscated then symetrically encrypted with a different key. The sequence of segment is messed up and only the execution of a segment  $S_i$  gives the decryption and the execution of  $S_{i+1}$ .



**Fig 7 Structure of a segment  $S_i$  of a C-Agent**

### Signed Stamps:

To protect an agent on a visited host we must limit the agent execution time. Also the amount of time in which the agent is active on a certain host must be precised to limit the agent responsibility in the case of an attack against that host. Thus we suggest the mechanism of signed stamps (fig 7) which consist in the association of the hash of the agent code, time (date) of the current host and the next host time (date) and the will all be signed by the two hosts.

### V. Conclusion

In this paper we presented the state of the art in securing mobile agents and an approach which we suggest. In the state of the art we identified strong and high cost approaches based on the use of additional hardware and weak and low cost approaches based on software solutions. The purpose of our approach is to provide a software based solution as strong as hardware based solutions. So, we propose a software machine that we call SVM (Secure Virtual Machine) which can interpret an agent without decrypting it. A mobile agent will be written in ordinary language then transformed by a software which we call the agent transformer to another agent (C-agent: ciphered agent) equivalent in function but difficult to analyse and executable only by the SVM. In order to protect a C-agent which is difficult but not impossible to analyse, we suggested the mechanism of signed stamps which limits the agent execution time on a visited host. In perspective, we are interested in securing SVM itself.

### IV. References:

- [FGS96a] : Farmer, William; Guttmann, Joshua; Swarup, Vipin: "Security for Mobile Agents: Issues and Requirements", in Proceedings of the National Information Systems Security Conference (NISSC 96), 1996.
- [Hoh 97] : Fritz Hohl, "An approach to solve the problem of malicious hosts", [www.informatik.uni-stuttgart.de](http://www.informatik.uni-stuttgart.de)
- [Wil 98] : <http://lsewww.epfl.ch/~wilhelm/thesis/>
- [Sah 98] : N. Sahli, "A synthesis of the state of the art in Internet Security and guidelines for developing countries", Africom 1998.
- [RFC2401] : S. Kent, R. Atkinson, "Security architecture for the Internet Protocol", [www.ietf.org](http://www.ietf.org), Novembre 1998.
- [Res98] : E. Rescola, A. Terisa, "The secure Hypertext Transfer Protocol", Internet Draft, [www.ietf.org](http://www.ietf.org), December 1998.
- [Netscape] : <http://www.netscape.com/>
- [Man02] : Antonio Maña, Ernesto Pimentel, "An efficient software protection scheme", University of Málaga, Spain, 2002.
- [VIG98] : G. Vigna. Cryptographic traces for mobile agents. In "Mobile Agents and Security" [Vig98b], pages 137{153.
- [FIPS180] : Federal Information Processing Standard, 'Secure Hash Standard' FIPS PUB 180, Mai 1993.
- [ST98] Tomas Sander and Christian Tshudin. Towards mobile cryptography. Proceeding 1998 IEEE symposium on security and privacy, pp. 215-224, Oakland, California, May 1998.
- [Lau 01] Sergio Laureiro, Mobile code protection, Thesis of doctoral degree, Institute of Eurocom, January 2001.
- [Jan00]: Wayne Jansen, countermeasures for mobile Agent security, Computer communication , Special issue on advance security techniques for network protection, Elsevier Science.
- [Jan01] : Wayne Jansen, Determining Privileges of Mobile Agents, Proceedings of the Computer Security Applications Conference, December 2001.
- [Jan2 01] : A Privilege Management Scheme for Mobile Agent Systems, First International Workshop on Security of Mobile Multiagent Systems, Autonomous Agents Conference, May 2001. Wayne Jansen.
- [Jan3 01] : Countermeasures for Mobile Agent Security, Wayne A. Jansen, National Institute of Standards and Technology, Gaithersburg, MD 20899, USA, October 2001.
- [ISO9594-8] : ITU-T Recommendation X.509 | ISO/IEC 9594-8: Information Technology - Open Systems Interconnection - The Directory: Public Key and Attribute Certificate Frameworks, March 2000.