A New Approach based on Cryptography and XML Serialization for Mobile Agent Security

Hind Idrissi[1,2], Arnaud Revel[1] and El Mamoun Souidi[2]
[1] *L3I Laboratory, La Rochelle University, La Rochelle, France*
[2] *MIA Laboratory, Mohammed V University, Rabat, Morocco*

Keywords: Mobile Agent, Multi-agent System, Mobility, Security, Cryptography.

Abstract: Mobile agents are a special category of software entities, with the capacity to move between nodes of one or more networks. However, they are subject to deficiency of security, related particularly to the environments on which they land or other malicious agents they may meet on their paths. Security of mobile agents is divided into two parts, the first one relates to the vulnerabilities of the host environment receiving the agent, and the second one is concerning the malevolence of the agent towards the host platform and other agents. In this paper, we will address the second part while trying to develop an hybrid solution combining the two parts. A solution for this security concern will be presented and performed .It involves the integration of cryptographic mechanisms such as Diffie-Hellman key exchange for authentication between the set (platform, agent) and the Advanced Encryption Standard (AES) to communicate the data with confidentiality. These mechanisms are associated with XML serialization in order to ensure easy and persistent portability across the network, especially for non permanent connection.

1 INTRODUCTION

Mobile agents are a particular class of software agents characterized by their ability to move between nodes of one or more networks. They are applied in Telecommunications (D. Gavalas and Anagnostopoulos, 2009), Internet with E-commerce (Fasli, 2007), optimization of transport systems (Chen and Cheng, 2010), management courses, and many other fields. The mobility of these agents could be strong or weak depending on the elements involved in the transfer process (code, data, stack, heap, counter, etc.), see (Ferber, 1999). Many systems for mobile agent have been developed in recent years such as Aglets (D.B. Lange and Kosaka, 1997), AgentTcl (Gray, 1997), Telescript (J. White, 1995) and JADE (F. Bellifemine and Rimassa, 2001).

The mobility over network's nodes is a very requested characteristic for multi-agent systems (MAS). It addresses the various deficiencies of the client/server paradigm (Gray and al, 2001), such as: Waiting time, Network traffic, Permanent connection, Transmission of intermediary and not useful information by maintaining properties of autonomy, efficiency, persistence, communication, adaptability and tolerance fault. However, this mobility is not all the time safe. A mobile agent can be attacked when it requests services from other servers or when it comes in contact and exchange information with another agents while roaming on the internet. These attacks may occur due to the unavailability of three security aspects: Authentication, integrity and confidentiality.

This paper attempts to address the security issues related to the mobility of agent. In Section 2, we describe the threats that the agent may meet along its migration. Then, in Section 3 we present our solution to correct these vulnerabilities using divers mechanisms, such as Diffie-Hellman key exchange protocol associated with digital signature to authenticate the communicating entities and ensure integrity of the migrating agent, AES encryption algorithm to guarantee the confidentiality of data exchanged, and the principle of XML serialization to obtain a persistent and transportable format of agent. Therefore, an implementation of the solution is proposed in Section 4, using JADE (F. Bellifemine and Rimassa, 2001) the Java Agent Platform designed for multi-agent systems and with respect to specifications of FIPA standard (S. Poslad and Hadingham, 2000) specified for software interoperability among agents and agent-based applications. While evaluating the feasibility and effectiveness of our solution in preventing attacks, we

Idrissi H., Revel A. and Souidi E..
A New Approach based on Cryptography and XML Serialization for Mobile Agent Security.
DOI: 10.5220/0004812004030411
In *Proceedings of the 6th International Conference on Agents and Artificial Intelligence* (ICAART-2014), pages 403-411
ISBN: 978-989-758-015-4

implement a baseline experiment consisting in a simple migration based only on XML serialization concept. This will allow to compute the overhead of security added to agent mobility. Finally, further steps and perspectives are discussed in conclusion.

2 SECURITY PROBLEM

Mobile agent technology brings significant gains for several application areas. Its strength to resolve complex problems is due to the fact that the agents with their autonomy, mobility and adaptability, can realize their goals in a flexible way by using a local or remote interaction with other agents on the network. However, the mobility of agents may raise security problems. When an agent moves from one site to another in order to be executed on each of them, it is crucial to ensure that the agent will run properly and safely on the new system visited. Similarly, it is important to guarantee to the host system that there will be no risk to host a new agent. In this section, we define the different vulnerabilities that mobile agent or platform may encounter. Let's consider the case of an application that requires the communications between n machines (noted M_i) connected by the TCP/IP network as illustrated in Figure 1 (with n=4).

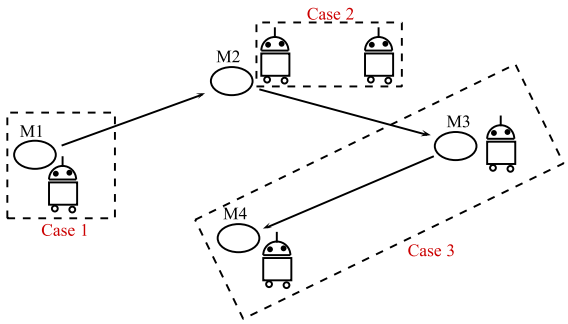

Figure 1: Cases of threat while mobility of agent between network nodes.

During its visits, the mobile agent may encounter many situations where its security is compromised. In Figure 1 three cases are outlined. The first case illustrates the interaction between mobile agent and host, this contact could be unsafe especially if the agent is unauthorized to access the platform, or if this later has the ability to manipulate the migrating agent. In the second case, the agent may interact along its path with other agents that can be malicious, and try to analyze or alter its contents. The third case describes the insecure exchanges between hosts and/or agents over the network. According to a review on mobile agent security (Ahuja and Sharma, 2012), three interactive situations requires security:

- *Case 1 - Agent affecting Host*: The mobile agent could have free and unauthorized access to the runtime environment and thus, it could violate its confidentiality, integrity and availability by intercepting or modifying its data, fully exploiting its resources, cloning or migrating indefinitely.
- *Case 2 - Agent affecting Agent*: An agent may undergo several attacks from other agents. We distinguish in this regard two types of attacks: passive attacks that do not change the code and data of the agent, and active attacks that could alter the code or data of the agent, by changing the variables values or inserting a virus.
- *Case 3 - Host affecting Agent*: It's the most complicated and difficult situation. When the agent is migrating to a host, it is forced to disclose its code, status and data, which makes it susceptible to confidentiality and integrity threats from the host, that exploits its information and manipulate its behaviors and results.

In this paper, an interest is given to the third case of Figure 1 where an agent could be damaged by a hosting platform. The complexity and the dynamic nature of interactions between agents and hosts make it difficult to predict the behavior of both. Thus, mobile agents carrying sensitive information about their owners should be protected from tampering by malicious hosts. This category of threats includes (Jansen and Karygiannis, 1998)

- Alteration: it occurs when an agent suffers from lack of its data integrity. An agent landing on a host must expose its code, state, and data to the platform. If this latter has a malicious behavior it may modify in the agent without being detected, which does not yet have a solution.
- Eavesdropping: a host can intercept and monitor secret communications, instruction executed by the agent, clear and public data as well as all the subsequent data generated on the platform.
- Masquerade: a host can masquerade as another host in order to deceive a mobile agent as to its true destination and then extract its sensitive information. The masquerading host can harm both the visiting agent and the host whose identity it has assumed.
- denial of service: A malicious host may ignore agent service requests such as not executing its code or introducing unacceptable delays for critical tasks, which could lead the agent to be deadlocked.

To establish a well-defined security policy for the threats mentioned in the previous paragraph, we need to satisfy the security requirements related to this paradigm: authentication, confidentiality, integrity and availability. A simple solution would be that the owner of an agent limits its itinerary only to trusted sites. However, this solution remains insufficient regarding the threats that agent may meet. Also, limiting the itinerary of agent will reflect passively on its mobility as well as the results it got. In this paper, many mechanisms are used such as: cryptography, mutual authentication, digital signature and XML serialization.

3 OUR SOLUTION

A mobile agent must have the ability to communicate with other agents of the system (either local or remote agents), to exchange information and benefit from the knowledge and expertise of other agents. But in practice, mobility does not replace the communication capabilities of the agents but completes them. Hence, the interaction between mobile agents needs first to initiate communication between the platforms, ensure their compatibility and collect specific information about them. In this section we present a detailed description of the proposed solution that consists in simulating a set of cooperative agents in charge of performing different mechanisms in order to satisfy the security requirements.

3.1 Authentication Solution

To prevent attacks related to unavailability of authentication, we integrate in each one of the interacting platforms, a specific agent called "$DH-DSA\_Agent$". This later must have among its data a specific list containing the addresses of hosts constituting the itinerary to travel. In practice, these addresses can be IP addresses of host machines. Before the migration of agent to a new host, an authentication mechanism using the Diffie-Hellman key exchange (Diffie and Hellman, 1976), and the standard for digital signature (Gallagher, 2009) is running between the "$DH-DSA\_Agent$" of both platforms, in order to create a common shared key. This key will be used afterwards to sign and verify the addresses and data exchanged between both hosts.

The first step of Diffie Hellman algorithm is to generate randoms for modulo and primitive root computations. This implies the use of Pseudo Random Number Generator (PRNG) to apply this task. Yet, the DH-provider in Java Runtime does not support cryptographic generator, considered as the faster and most secure ones after the quantum generators. This issue leads us to adopt a new implementation of Diffie Hellman algorithm using ISAAC+ (Aumasson, 2006). The ISAAC+ algorithm is an enhanced version provably secure of ISAAC (Indirect, Shift, Accumulate, Add and count) (Aumasson, 2006) which has similarities with RC4 (Mousa and Hamad, 2006). it uses an array of 256 four-octet integers as the internal state, and writes the results to another 256 four-octet integer array. It is very fast on 32-bit computers.

In our approach, we make use of a new attempt to fix the integrated Diffie-Hellman-DSA Key Exchange Protocol proposed in (Phan, 2005). Figure 2 enumerates the different steps of the improved protocol. All random values are generated with ISAAC+, the computations are performed on finite field, and for the digital signature we use the one-way function SHA-1 (Eastlake and Jones, 2001). At the step 10, we introduce the IP address of the remote host (got from the list of addresses that the mobile agent contains) in the signature, and at the step 11 the hosting platform verifies that signature using its own IP address.

The main idea behind this attempt to fix the integrated Diffie-Hellman-DSA Key Exchange Protocol is to ensure computations basing on two ephemeral secrets v and w chosen by the two parties A and B. This provides forward secrecy because even if the long-term private key of any party is exposed, previous session keys cannot be computed since the ephemeral secrets, v and w for that session are unknown. Also provides key freshness because every session key is a function of ephemeral secrets so neither party can predetermine a session key's value since he would not know what the other party's ephemeral secret is going to be.

Figure 3 describes the process of the adopted solution for authentication. The native machine sends to the host one a request for mobility of agent and asks for information to authenticate it. Each machine includes a manager agent responsible for managing communications between components of the platform and interactions with the remote ones. This manager agent communicates with the "$DH-DSA\_Agent$" in order to perform the steps of the authentication protocol and generate a session shared key of 256 bits used later to maintain the confidentiality and integrity properties .

The establishment of an authentication mechanism between agents and platforms is very essential, to avoid attacks in relation with unauthorized access. An agent that has access to a platform and its services without having the proper authorization can harm other agents and the platform itself. So, a plat-

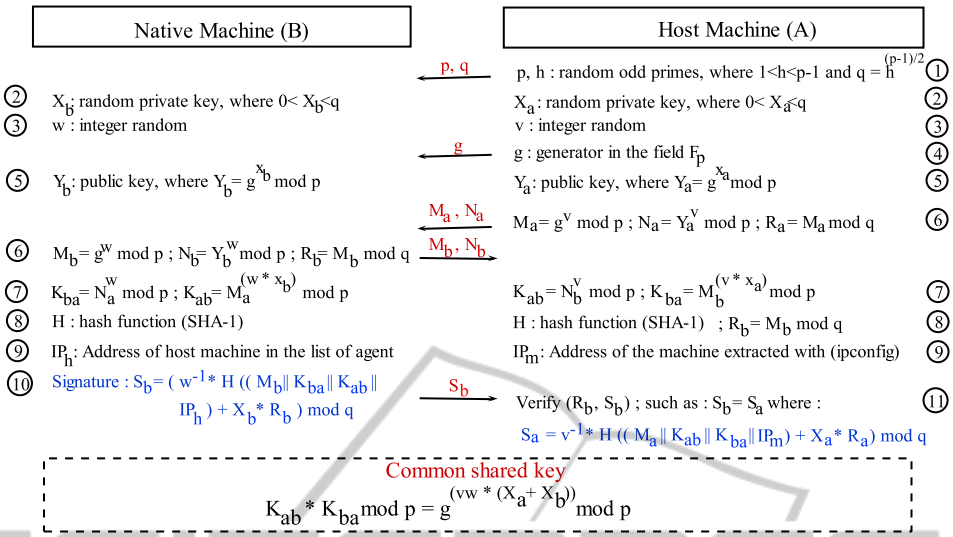

Figure 2: Integrated improved DH-DSA Key Exchange Protocol.

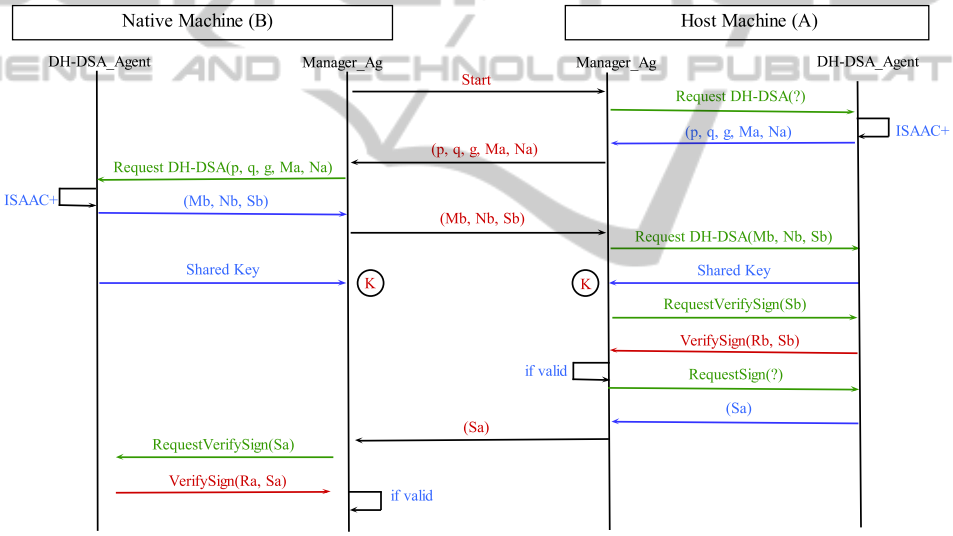

Figure 3: Authentication solution.

form that hosts agents must ensure that agents do not have access if they have no authorization. To apply a proper and well-defined access control, the platform or the agent must first authenticate the mobile agent's identity, before it is instantiated on the platform. There are several methods for mutual authentication on the net, among them the " HTTPS Mutual Authentication ". This later combines to the protocol secured layers using encryption such as TLS (Transport Layer Secure) to generate secured passwords (TLS-EAP) and to verify the identity of the client and server using Certificates. Thus, many attacks are avoided: Man-in-the-middle attack, offline password dictionary attack and phishing. However, this mechanism was recently broken as it suffers from serious deficiencies, like layering problem as it is impossible to match authentication session and transport session, and certificates management problem. In our solution we don't make use of certificates, so we gain in terms of resources dedicated to implant a Certificate Authority (CA) and in terms of time consumed to interact with the CA and charge certificates. Also, in both platforms, there is a distribution of tasks to the agents with a tracing of executed ones which creates a matching between different level of execution. Besides, the protocol we used is based on complex problems in mathematics such as discrete logarithm.

3.2 Integrity Solution

In the previous subsection, a shared key was generated for both interacting platforms, and a digital signature algorithm was established. Therefore, to preserve the integrity of exchanged data, both hosts must send to each other a signature, and verify the received one. This mechanism is described in last steps of Figure 2. After creating a common key (T), the native machine signs the set of keys generated by the protocol with the IP address of the remote host, and then sends this signature to the relevant host. This later in turn signs the keys generated with its IP address, and sends the signature to the native machine. Both machines must verify and validate the signature to carry on processing.

The unavailability of the integrity may expose the agent to several threats such as "Alteration attack". When an agent migrates to an agent platform it exposes its code, state, and data to the platform, which can modify its carried contents of information. Indeed, preventing the platform from modifying the agent is strongly needed, as well as detecting changes which does not yet have a general solution. To provide integrity feature for our approach, Digital Signature Algorithm (DSA) is used to sign data exchanged.

3.3 Mobility Solution

Mobility in the context of transportability across the network can be performed through layer protocols such as HTTP provided with the Apache server, also it can be done by JAVA APIs like RMI (Remote Method Invocation) or RPC (Remote Procedure Call) that manipulate remote objects. These APIs are structured in network layers based on the OSI model to ensure interoperability between programs and versions of Java.

For mobility feature of our approach, we adopt a weak mobility where agent restarts its execution for each visited host, and we make use of the XML serialization mechanism that provides a persistant and well transportable data across the network. This association is the most appropriate for our approach and it addresses the flaws of other modes of transportability, it is easy to develop and ensure convenience and efficiency as it generates an easily readable and editable format. Figure 4 describes the mobility process using this mechanism. In the native machine, an agent named "*Serial_Agent*" is implemented in order to serialize in XML format an instance (object) of the mobile agent class. This class contains the attributes and execution code of the agent. Then, the XML instance given is encrypted by "*AES_Agent*" using the session key and transferred to the host machine. At the other side, the class of mobile agent is rebuilt using the encrypted XML instance of the agent, which is firstly decrypted by "*AES_Agent*" using the session key, and de-serialized using the XMLDecoder of JavaBeans API.

The problem encountered in this simulation, was when the host platform does not support the JavaBeans of XML serialization or does not recognize certain classes needed in execution. After research, we have found that the principle of URLClassLoader is the most appropriate for our approach. It will give us the possibility to load the classes and packages needed for execution from the URL of native machine or any other databases in the network, which implants an aspect of availability of information.

3.4 Confidentiality Solution

When information must be transmitted between two systems, especially heterogeneous (D. Mitrovic and Vidakovic, 2011) by their security, it can be intercepted or modified by an intruder. In order to preserve confidentiality we based primarily on the use of Cryptography. There are two categories of cryptographic systems: secret key cryptosystems and public key cryptosystems. In the authentication part, a session key was created and shared between the two platforms, also a pair of public and private keys are generated due to digital signature process. Hence, we can choose the category judged adequate. Secret key cryptosystems are considered effective as they rely on complex methods, but they are very slow and consume more computer resources to perform the computations and store the generated keys that are mostly very long. However, the public key cryptosystems use less memory and resources with small key length. They are much faster in processing and largely used for data compression, but they need secure channel to share the secret key.

The solution adopted for authentication solves the secret key cryptosystem problem of securing the key exchange, so the choice of this cryptosystem is justified by the small length of keys and fast computations. The algorithm adopted is AES256: Advanced Encryption Standard (Robertazzi, 2012), which is robust and introduces a key length of 256 bits that matches with the length of the key session obtained. In Figure 4, the "*Manager_Ag*" sends the serialized object of the agent class with the session key to the "*AES_Agent*". This later takes in charge the encryption of the serialized object on the native machine, and its decryption on the host machine.

The lack of confidentiality in data exchange with a platform may expose the agent to the "Eavesdrop-

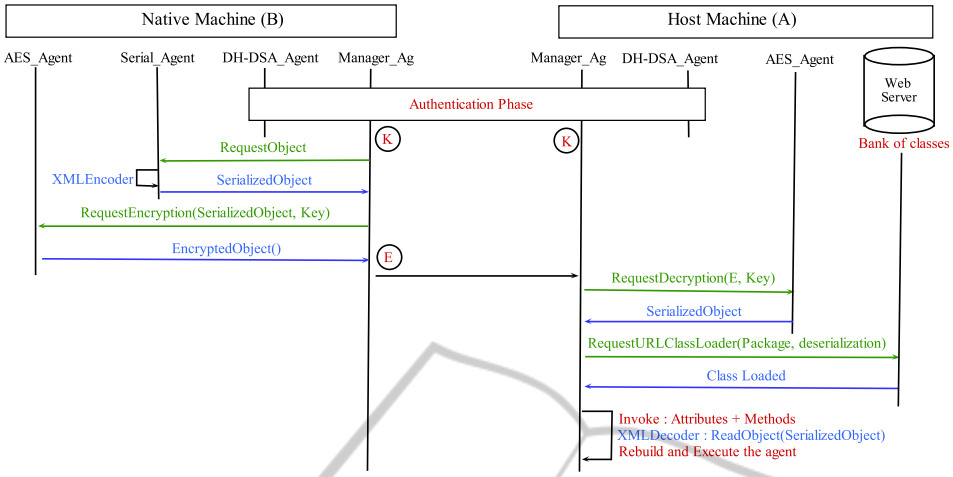

Figure 4: Confidentiality solution.

ping Attack". However, this threat is further extended because the agent platform can not only monitor communications, but also can monitor every instruction executed by the agent. Thus, we use the secret key system AES to encrypt and decrypt data at the both sides. This avoids any intruder to know the real content of the message exchanged even it got it. In other words, this supports the secrecy of exchanging.

4 OVERHEAD

In this section we present the results of experiments performed by running an implementation of the proposed solution, using JADE agent framework. In the previous section, while giving a description of the solution, we have analyzed its ability to prevent attacks such as: "Unauthorized Access Attack", "Alteration Attack" and "Eavesdropping Attack". Thus, our evaluation efforts have been focused on testing the time spent by migrating agents, under the conditions mentioned above. This section is divided into two parts: Theoretical Analysis that presents a formal computation of time spent for interactions of our solution, and Experiments in which two tests are performed to calculate the overhead of security provided. The first one is considered as basic test that illustrates a simple migration process without integrating the security mechanisms discussed, and the second one is an implementation of the solution proposed.

4.1 Theoretical Analysis

Normally, a launched mobile agent migrates from one site to another and then comes back to the first one, in what it is called a round-trip. Figure 5 shows the stages of the agent round-trip:

- In the going, the agent is involved in an authentication process that uses ISAAC+ and Diffie-Hellman-DSA protocol which generates a session key. Afterwards, the agent is serialized in an XML object format and encrypted with AES256 algorithm using the session key.
- The host receives the serialized encrypted object and tries to rebuild the agent by first decrypting the object using AES256 and then deserializing it. When the class of the agent is unreachable we make use of URLClassLoader. Once the agent rebuilt, it is executed and an acquittal is sent to the native machine in order to delete the agent.
- In the return, the agent brings results of execution to its native machine.

Let's consider $T_{rt} = Time$ of the round trip, such as:

$$T_{rt} = T_1 * N_{jump} \times 2 \quad (1)$$

Where: N_{jump} = the number of jumps during the execution, it is based on the data exchanged in each stage. T_1 is a period comprising several sub-periods related to solution process stages:

$$T_1 = T_{isaac} + T_{dh-dsa} + T_{serialization} + T_{encryption} + T_{decryption} + T_{deserialization} + T_{requests} + T_{datasending}. \quad (2)$$

Knowing that $T_{serialization}$ is approximately equal to $T_{deserialization}$, and $T_{encryption}$ is approximately equal to $T_{decryption}$. Then: $T_{serialization} + T_{deserialization} = 2T_{serialization}$ $T_{encryption} + T_{decryption} = 2T_{encryption}$ Hence, the equation 2 becomes:

$$T_1 = T_{isaac} + T_{dh-dsa} + 2T_{serialization} + 2T_{encryption} + T_{requests} + T_{datasending}. \quad (3)$$

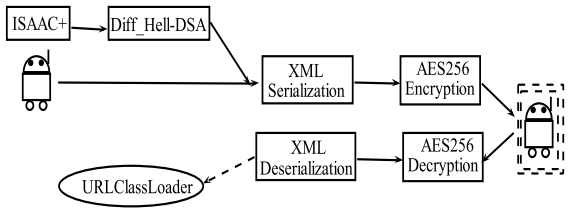

Figure 5: Round trip of agent mobility.

Since the request messages are exchanged in intra-platform and their length does not exceed four characters, then no computation time is consumed for initiating communication, no memory is allocated and no frames for transporting data across the network. Hence, the time of requests could be negligible. So equation 3 becomes:

$$T_1 \simeq T_{isaac} + T_{dh-dsa} + 2T_{serialization} + 2T_{encryption} + T_{datasending}. \quad (4)$$

The time of one round trip is calculated according to equation 1. However, during the going of mobile agent there is an added time, which is the time of loading the class using URLClassLoader, when the host does not recognize it. In the return to native machine, the agent doesn't need to load the class because it is already provided. Finally a round trip takes the time:

$$T_{rt} = (T_1 + T_{loadingclass}) \times N_{jump} + T_1 \times N_{jump} \quad (5)$$

4.2 Experiments

Our experiments are performed using 4 heterogeneous machines. The first one is considered as a native machine and others as hosts. All machines are Core i7 at 2.7 GHz, with 4 Go of RAM, 500 Go on an ATA 500 hard disk, and are equipped with Windows operating system in the XP version, and used also a 100Mbps switched Ethernet network with WampServer. For agent execution, JADE Snapshot agent platform is used in its 3.6.1 version.

4.2.1 The Baseline Test

In order to evaluate the feasibility and performance of our solution, we find it interesting to configure a baseline test, that consists in performing a simple mobility of agent using only XML serialization concept without trying to encrypt it or to authenticate it. This allows to calculate the overhead needed for securing the mobile agent. The results of this test are exposed in the table 1.

In such normal conditions where the agent is only serialized and transferred, the equation 4 becomes:

$$T_1 = T_{serialization} + T_{datasending}, \text{ with } N_{jump} = 2 \quad (6)$$

Table 1: Performance test with one native machine and several hosts.

Time(ms)	1 Host	2 Hosts	3 Hosts
$T_{serialization}$	**73**	**91**	**143**
$T_{datasending}$	**127**	**243**	**339**
$T_{loadingclass}$	**338**	**914**	**2463**

Referring to results in the table 1 and according to equations 5 and 6, the cost for mobility of an agent to one host, in normal conditions, can be calculated as follow:

$$T_{rt} = (((2 \times 73) + 127 + 338) \times 2) + (((2 \times 73) + 127) \times 2) = 1222 + 546 = 1768ms$$

In theory, the time to move an agent over three hosts is three times the time of moving an agent to one host. However, the use of different and heterogeneous machines for experiments, makes our results not uniform. In order to know the overall difference, we compute the average of time that agent spent while migrating over three hosts:

$$T'_{rt} = \frac{(((2 \times 143) + 339 + 2463) \times 2)}{3} + \frac{(((2 \times 143) + 339) \times 2)}{3} = 2058 + 413 \simeq 2470ms$$

Then, the difference of time is about: $D_{time} = T'_{rt} - T_{rt} = 2470 - 1768 = 702ms$. The large part of this difference concerns the time of loading classes over network and for heterogeneous machines.

4.2.2 Implementation of our Solution

The second experimentation consists in running the implementation of our solution, taking into consideration the four aspects mentioned before. This test launches the set of agents representing the different mechanisms used to reach a high level of security. The results we got through running this second test are shown in the table2.

Referring to these results, and according to equations 4 and 5 with $N_{jump} = 2$, the time spent when moving one agent under the circumstances of our solution is:

$$T2_{rt} = ((2.4 + 6.4 + (2 \times 78) + (2 \times 15) + 343 + 182) \times 2) + (((2.4 + 6.4 + (2 \times 78) + (2 \times 15) + 182) \times 2) = 1439.6 + 753.6 = 2293.2ms$$

Similarly to the first test, it was taken in charge to calculate the difference of time between agent migration to one host and the average of agent migration to

Table 2: Performance test with one native machine and numerous hosts.

Time(ms)	1 Host	2 Hosts	3 Hosts
T_{isaac}	**2.4**	**3.9**	**6.1**
T_{dh-dsa}	**3.8 (parameters) + 2.6 (computations) = 6.4**	**20.8**	**48**
$T_{serialization}$	**78**	**94**	**141**
$T_{encryption}$	**15**	**33**	**50**
$T_{datasending}$	**182**	**318**	**432**
$T_{loadingclass}$	**343**	**907**	**2470**

three hosts:

$$T2'_{rt} = ((6.1+48+(2\times141)+(2\times50)+2470+432))\times\frac{2}{3}+\frac{(((6.1+48+(2\times141)+(2\times50)+432)\times2)}{3} \simeq 2225+578\simeq 2803ms$$

Then, the difference of time is about: $D_{time} = T2'_{rt} - T2_{rt} \simeq 2803-2293\simeq 510ms$.

Finally, we can extract from both experiments the overhead dedicated for securing the mobility of agent. This overhead is related mainly to the authentication and integrity by the key exchange associated with digital signature, also to the confidentiality by the encryption mechanism. The computational value of this overhead is:

$$T_{solution} - T_{baseline} = 2293.2 - 1768 \simeq 525ms.$$

This value represents 28% of overall time that agent takes to move. It involves the cost of authentication process, including the generation of ISAAC+ randoms, cryptographic keys and digital signatures. It includes also the cost of the AES encryption / decryption process with key of 256 bits, the cost of loading classes using URLClassLoading and the cost of exchanging data between hosts and agents. The overhead of 525 ms is considered as admissible, credible and not compromising for the mobility performance of the agent, which benefits from a security feature protecting him against vulnerabilities of hosts.

5 CONCLUSIONS

This paper presents some initial results of a research effort aimed at the analysis of the security issues in mobile agent systems. It also describes a proposed solution to elaborate a security policy using mechanisms such as cryptography, digital signature and XML serialization. These mechanisms are mainly introduced to ensure the properties of confidentiality, integrity, authentication and give an enhanced mobility for the agent migrating from one site to another across network. The results of agent performance using this solution, and the security analysis provided by evaluating its effectiveness to prevent attacks such as: "alteration attacks", "Man-in-the-middle attacks" and "Eavesdropping attacks", can be reused as guidelines to develop secure mobile agent systems involved in mission-critical applications.

The future work will focus on completing the security analysis of the mobile agent systems, and in developing a reference security model using mechanisms and methods that proved their effectiveness without affect the sensibilities of mobile agents. Among our perspectives, we will try to more enhance the use of URLClassLoader and attempt to find a solution for the communications under a non permanent connection, as well as optimizing the mobility of agent following its itinerary using optimization algorithm. We will also take an interest of the other categories of threats, especially those in relation with harming agent platforms. We will be concerned with developing a well-defined access control policy to the resources of the platform. In security context, we will try to find a solution of denial of service attacks using detection methods.

REFERENCES

Ahuja, P. and Sharma, V. (2012). A review on mobile agent security. *International Journal of Recent Technology and Engineering (IJRTE)*, pages 2277–3878.

Aumasson, J. (2006). On the pseudo-random generator isaac. *IACR Cryptology ePrint Archive*, 2006:438.

Chen, B. and Cheng, H. (2010). A review of the applications of agent technology in traffic and transportation systems. *Intelligent Transportation Systems, IEEE Transactions on*, 11(2):485–497.

D. Gavalas, G. T. and Anagnostopoulos, C. (2009). A mobile agent platform for distributed network and systems management. *Journal of Systems and Software*, 82(2):355–371.

D. Mitrovic, M. Ivanovic, Z. B. and Vidakovic, M. (2011). An overview of agent mobility in heterogeneous environments. In *Workshop Proceedings on Applications of Software Agents*, page 52.

D.B. Lange, M. Oshima, G. K. and Kosaka, K. (1997). Aglets: Programming mobile agents in java. In *Worldwide Computing and Its Applications*, pages 253–266. Springer.

Diffie, W. and Hellman, M. (1976). New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644–654.

Eastlake, D. and Jones, P. (2001). Us secure hash algorithm 1 (sha1).

F. Bellifemine, A. P. and Rimassa, G. (2001). Jade: a fipa2000 compliant agent development environment. In *Proceedings of the fifth international conference on Autonomous agents*, pages 216–217. ACM.

Fasli, M. (2007). *Agent technology for e-commerce*. John Wiley & Sons Chichester.

Ferber, J. (1999). *Multi-agent systems: an introduction to distributed artificial intelligence*, volume 1. Addison-Wesley Reading.

Gallagher, P. (2009). Digital signature standard (dss). *Federal Information Processing Standards Publication, FIPS PUB*, pages 186–3.

Gray, R. (1997). Agent tcl: A flexible and secure mobile-agent system.

Gray, R. and al (2001). Mobile-agent versus client/server performance: Scalability in an information-retrieval task. In *Mobile Agents*, pages 229–243. Springer.

J. White, J. (1995). Telescript technology: An introduction to the language. *General Magic White Paper, General Magic*.

Jansen, W. and Karygiannis, T. (1998). Mobile agent security. Technical report, National Institute of Standards and Technology.

Mousa, A. and Hamad, A. (2006). Evaluation of the rc4 algorithm for data encryption. *IJCSA*, 3(2):44–56.

Phan, R.-W. (2005). Fixing the integrated diffie-hellman-dsa key exchange protocol. *Communications Letters, IEEE*, 9(6):570–572.

Robertazzi, T. (2012). Advanced encryption standard (aes). In *Basics of Computer Networking*, pages 73–77. Springer.

S. Poslad, P. B. and Hadingham, R. (2000). The fipa-os agent platform: Open source for open standards. In *Proceedings of the 5th International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents*, volume 355, page 368.