International Journal of Security and Its Applications
Vol. 9, No. 5 (2015), pp. 19-30
http://dx.doi.org/10.14257/ijsia.2015.9.5.02

Mobile Agent Security Using ID-Based Agreement Protocol and Binary Serialization

Hind Idrissi[1,2], El Mamoun Souidi[1] and Arnaud Revel[2]

[1]*Laboratory of Mathematics, Computing and Applications, University of Mohammed-V, Rabat, Morocco*
[2]*L3I Laboratory, University of La Rochelle, France*
hind.idr@gmail.com, emsouidi@fsr.ac.ma, revel.arnaud@univ-lr.fr

Abstract

Recent days, the systems based on mobile agents have attracted the attention of many researchers in different areas, because of the autonomic and pro-active aspects of the agent, as well as its adaptive and dynamic behaviors in solving complex problems. However, the mobility of these entities needs to be secured against multiple vulnerabilities that present a real obstacle to its wide expansion. When the mobile agent is migrating from one node to another over the network, it is difficult to guarantee that it will be executed safely and correctly on the hosting platform, neither that it will not encounter in its way malicious entities that try to harm it. In this paper, we try to address these security issues by introducing an approach based on cryptographic mechanisms. This approach involves an Identity-Based Key Agreement Protocol to get a session key and ensure authentication, an Advanced Standard Encryption (AES) for the confidentiality of data exchanged, as well as a Binary Serialization to get an easy and persistent portability of the agent across the network.

Keywords: *Mobile Agent, Security, ID-Based Key Agreement Protocol, AES, Binary Serialization, JADE*

1. Introduction

Mobile Agents in network are software entities endowed with the capacity to move from one node to another of one or multiple networks. Actually, there is no standard definition for what an agent is, but the most common one is that of Ferber [1] that defines an agent as "a physical or virtual entity able to act in an environment, it possesses its own resources and gives services as individual objectives". Mobile Agents are characterized by their autonomy, adaptability and independence that make them not bound to the platforms where they are migrating. These qualities are due to the fact that they transport all their resources: code, data and state, which allow them to execute their tasks independently and adapt dynamically their behaviors according to specification of the environment.

The field of mobile agents becomes one of the more dynamic research areas in recent years. It is increasingly integrated in several evolving disciplines, such as personal information management [2], electronic commerce [3], information retrieval [4], telecommunications [5] computer games [6], industry [7], optimization of transport systems [8], and many other fields.

Certainly, the mobility of agents is very requesting and interesting characteristic, as it solves many problems met with the traditional Client/Server architecture, such as the increase of network traffic with the increase of users queries, the waiting time and response delays that the clients are obliged to support, the transmission of intermediary and not useful information and the necessity to perform under

ISSN: 1738-9984 IJSIA

permanent connections. However, this mobility raises many security issues as it is not all the time safe. During its migration, the mobile agent can be attacked when it requests services from malicious hosts or when it interacts and exchange sensible information with other harmful agents met in its itinerary.

In this paper, we try to address the security vulnerabilities related to the mobility of the agent. In Section 2, an overview is given about the different security threats that the agent may face along its mobility. Section 3 provides a detailed description of the proposed solution to deal with these vulnerabilities. This solution includes diverse mechanisms, such as ID-Based Key Agreement protocol to authenticate the interacting entities and generate a session key. This later is used afterwards to ensure the confidentiality of data exchanged along with AES encryption. A binary serialization of the mobile agent object or instance is integrated to guarantee a persistent and easy transportable format over the network. In Section 4, an implementation of the proposed solution is provided using JADE [9] the Java Agent Development Framework, and the results of the evaluation are provided and debated. Finally, further constraints and perspectives are discussed in conclusion.

2. Security Problem

Nowadays, the distributed systems strongly request a complex computing that can be performed online with a non-permanent connection. This ensures the availability of information and improves its efficiency. Mobile agent is one of the technologies dedicated for this process as it enables to perform all necessary computations with total independence of the environment where it is located. However, the mobility of the agent may be exposed to different security problems. When an agent moves across the network nodes in order to be executed on each of them, it is crucial to guarantee that it will operate properly and safely on the new system visited. Similarly, it is important to ensure to the hosting platform or environment that there will be no risk to execute a new agent. In the remainder of this section, the situations raising these security issues are presented, and the different threats that especially a mobile agent may encounter are discussed.

Figure 1 considers four communicating machines (M_n), linked with a connection over TCP/IP transmission protocol. During the trip of the agent, there are three arising cases where the security is compromised. The first one shows the unsafe interaction between a mobile agent and a malicious hosting platform. The second one illustrates an external malicious mobile agent trying to get access to a trusted platform, even it is not authorized to do, and the third case shows that the mobile agents may face in its path other malicious agents.

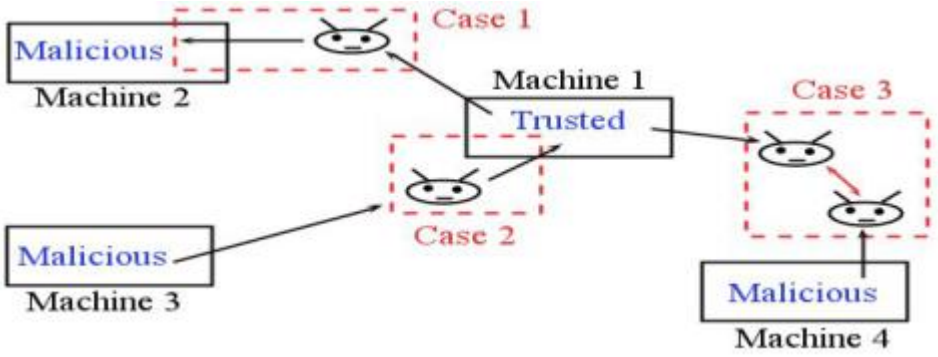

Figure 1. The Cases Arising Security Issues during the Migration of the Agent

According to [10], the three cases mentioned before are defined as follow:

- ***Case 1 – Platform against Agent***: When an agent is migrating to a new destination, namely new platform, it has to expose in clear its code, status and data, which makes it susceptible to confidentiality and integrity threats on behalf of the hosting platform that exploits its information and manipulates its behaviors and results.
- ***Case 2 - Agent against Platform***: A 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 3 - Agent against Agent***: An agent moving over the network may undergo several attacks from other agents encountered in the itinerary. In this regard, there are two defined types of attacks: passive attacks that just observe and analyze behaviors and results of the agent without changing its code and its data, and active attacks that could alter the code or data of the agent, by changing the variables values or inserting a virus.

In our paper, we are concerned by the first case, where the mobile agent can be at the mercy of a malicious hosting platform. In reality, it is very difficult to anticipate the behaviors of the mobile agents, neither those of the hosting platforms, because of the complexity of the dynamic interactions between the both. Thus, this case includes a set of attacks [10]:

- ***Masquerading***: A hosting platform can masquerade and claim the identity of another one in order to convince the mobile agent of being the true destination requested, and then extract its sensitive information. In this case, both the mobile agent and the platform whose identity was assumed are harmed.
- ***Alteration***: It occurs when an agent suffers from unavailability of data integrity. When a mobile agent reaches a platform, it must expose its code, state, and data to the platform. A platform with malicious behavior could modify or remove in the agent, or may destroy it without being detected.
- ***Denial of service***: A malicious platform may ignore a part or the entire services that the migrating agent is requesting, such as not executing its code or introducing unacceptable delays for critical tasks, which could lead the agent to be dead-lock.
- ***Eavesdropping***: A hosting platform 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.

3. Proposed Solution

The mobility of the agent and its capacity to roam freely over network nodes, does not substitute the fact that it must hold the ability to well communicate with its entourage, in order to exchange information and benefit from the knowledge and expertise. Thus, the first concern is to initiate the communication between the interacting entities, check their compatibility and authenticate them before exchanging sensible data end executing tasks. This section provides a deep description of the proposed solution that makes a set of agents interacting and cooperating in order to satisfy the security requirements of the system.

Let us first see the architecture adopted to simulate the solution. Figure 2 illustrates the set of agents integrated in the two interacting platforms. The “Admin_Ag” is the administrator agent charged with managing the communications of both platforms. Each incoming or outgoing request or information to the platform is controlled and analyzed by this agent. It has also the function to administer the communications between the local agents and direct the tasks associated with them. The “IKA_Ag” is the agent charged with performing the ID-based key agreement protocol and the authentication, the

"BinSerial_Ag" is the agent charged with carrying out the binary serialization and deserialization of the mobile agent object, the "AES256_Ag" is the agent charged with encryption and decryption of the mobile agent using the session key generated during the authentication, and the "Sign_Ag" which is responsible of editing a Schnorr signature for each data exchanged to check their integrity. The roles of these agents are well explained in the next paragraphs.

3.1. Authentication between Mobile Agent and Platform

Authentication is an important process that verifies the identity of another entity (person, computer...) to allow its access to resources (systems, networks, applications...), which permits then to validate the authenticity of the entity in question.

Authentication between the platforms is the first and essential resource before the execution of the agent, because a malicious platform could hide its identity in order to attract the agent and then leads attacks on it. In order to establish an authentication between the native platform of the mobile agent and the remote platform where it intends to migrate, a specific agent called "IKE_Ag" is integrated in the both. This agent is responsible for the modeling of an ID-Based Key Agreement protocol (IKA) [11], that aims to authenticate both platforms and generate a secret session key used then to encrypt all exchanges between them.

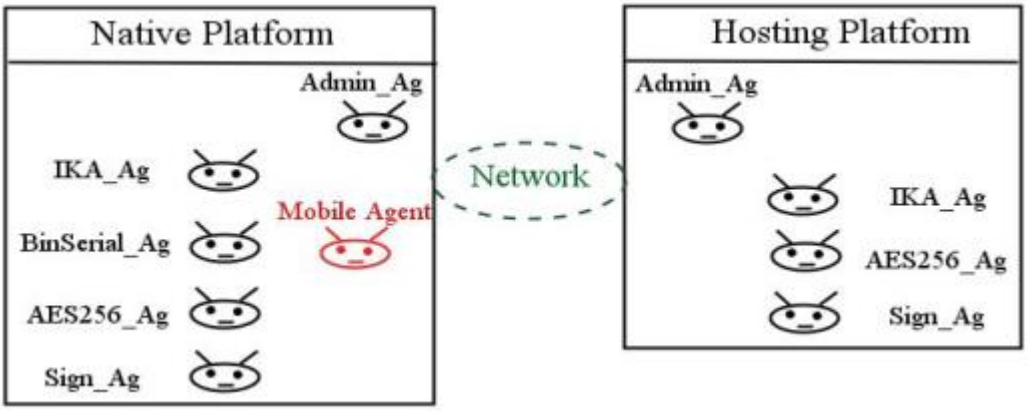

Figure 2. The Agent Architecture of the Communicating Platforms

An identity-based key-agreement protocol [11] is run by parties interacting in a network where each party is identified by a unique and secret identity (*e.g.*, Alice's identity is a string ID_A that includes form example a hash of concatenated ID of JADE platform with Username, IP address, MAC address and realm in randomized order). In addition there exists a trusted entity called Key Generation Center (KGC) that generates the public parameters of the system and also issues secret keys to users associated with their public identities. In our paper, we make use of an IKA protocol inspired from [12], and we enhance it with some adjustments to ensure the targeted authentication. This protocol is based on three phases: The setup phase, the key extraction phase and the key exchange phase.

Before explaining the authentication process, it is worth to mention that for simple and deterministic simulation, we consider that the "IKA_Ag" contains beforehand a list of the identifiers of the hosting platforms that the mobile agent intends to visit. This list of identifiers is retrieved from a Trusted Third Party, such that it corresponds to the confidence threshold and the constraints related to the execution and information sought decided by the native platform. But, this does not prevent the application of the same process for dynamic and adaptive mobility,

where the agent ignores where it has to migrate and takes a decision once it found attractive indexes in real-time.

The first step in the IKA-Authentication protocol is the generation of randoms, which implies the use of Pseudo-Random Number Generator (PRNG). For that purpose, we make use of ISAAC+ (Indirect, Shift, Accumulate, Add and Count) [13] a cryptographic generator considered as the faster and most secure ones after the quantum generators. It has similarities with RC4 and 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.

Figure 3 describes the process of the improved IKA-Authentication protocol. The random numbers are generated using ISAAC+ and the operations are performed on finite field. The native platform sends to the KGC the identity of the remote hosting platform that the mobile agent wants to visit, in order to authenticate it later. Otherwise, the hosting platform sends its own identity that it has to be verified and authenticated. At the extraction phase, each one of the interacting platforms receives from the KGC (Key Generation Center) a private key computed as the Schnorr's Signature [14] of the its identity string, under the public key *y*. Afterwards, at the key exchange phase, both platforms exchange credentials based on random exponents under the random generator of the group chosen. Among the credentials that the hosting platform sends, there is a hash value of e_H its own identity ID_H concatenated with r_N received. Therefore, In order to verify the identity of the remote host, the native platform computes the hash of the intended platform identity ID_R concatenated with a the value of V_H, and checks if it corresponds to value of e_H. Finally the session key is computed as follow:

$$Z = H_2(z_1, z_2) = H_2\left(g^{(t_H+S_H)(t_R+S_R)} g^{t_H t_R}\right)$$

Figure 4 illustrates the authentication process and the interactions between the agents of the two communicating platforms. First, the "Admin_Ag" of the native platform notifies the hosting platform with its will to move an agent, and advices it of the necessity to be authenticated.

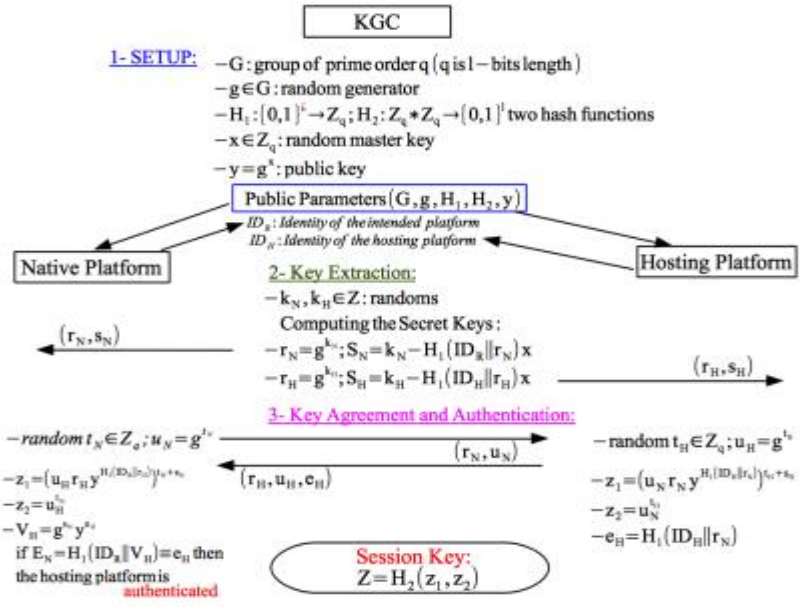

Figure 3. The Process of the Authentication ID-Based Key Agreement Protocol

Thus, the "Admin_Ag" of the hosting platform requests an authentication process, in which the IKA_Agents of both platforms and the KGC are integrated and a session key of 256 bits is generated. This key is used later in an encryption process to maintain the confidentiality and integrity of data exchanged.

In general, the authentication is an essential mechanism that avoids attacks based especially on unauthorized access. Definitely, there are several protocols and functions to ensure authentication, such as protocols that combine TLS (Transport Layer Secure) with Certificates Authority (CA). Even if these protocols avoid many attacks, but they are easily broken as it is impossible to match authentication session with the transport session and the certificates management problem. Not to mention the overhead, in terms of space allocation and time, to implant and communicate with a CA. Otherwise, The authentication-IKA protocol we adopt for this paper, is proven to be secure under the Strong Diffie-Hellman assumption (SDH), and provides many security properties [12]:

- ***Forward Secrecy***: After a session is completed and its session key is erased, the adversary cannot learn it even if it corrupts the parties involved in that session.
- ***Resistance to reflection attacks***: An adversary cannot compromise a session in which the two parties have the same identity (and the same private key).
- ***Resistance to Key Compromise Impersonation***: When the adversary learns the private key of an interacting platform, this does not allow him to impersonate this platform to other entities.

3.2. Binary serialization of persistent mobility

The systems based on mobile agents support two types of mobility: weak and strong. The weak mobility only transfers agent with its code, data and some special moments (breakpoint) in the agent code to be run.

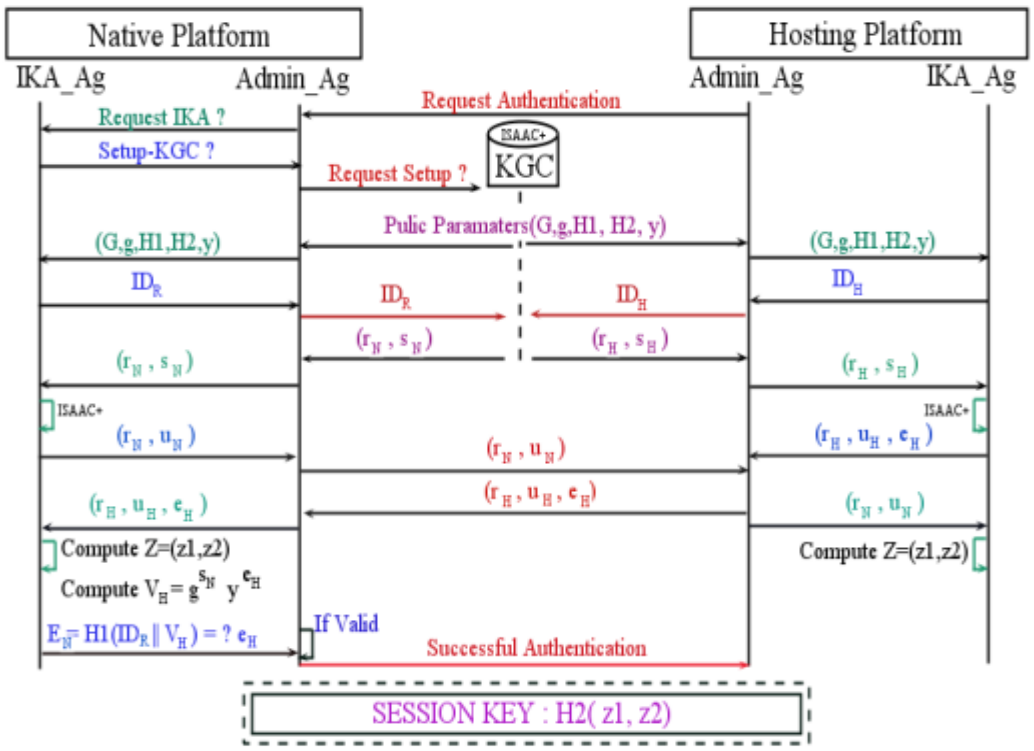

Figure 4. The Authentication Process between the Two Platforms

It may be initiated at any time but with loss of current treatment, *i.e.*, it resumes its execution from the beginning. The systems which support weak mobility, save and restore only the runtime values of (all or some of) the agent's properties. The agent's execution is resumed on the target machine by, for example, automatically sending a predefined message to the agent. On the other hand, the strong migration allows an agent to move across the network, regardless of the runtime state in which it is located, and to continue its execution after migration exactly where it was stopped before. It includes saving and restoring the entire runtime state of the agent, including the execution stack, the program counter, and so on. Strong mobility is more transparent than weak, in the sense that agent is completely unaware of it. However, it is technically more difficult to implement.

In practice, the mobility of agents can be performed using data transfer protocols based on TCP/IP, such as: HTTP, FTP, SMTP and RTP. These protocols are usually connected with a networking service such as Apache Server, where data are exchanged as sockets. Another possibility to achieve the mobility is to make use of JAVA APIs like RMI (Remote Method Invocation) and RPC (Remote Procedure Call) to monitor distant entities. In this paper, we adopt a weak mobility in a context of agent transportability across the network. Moreover, to enhance this transportability and makes it easy and persistent, it appears very interesting to make use of a binary serialization mechanism. This later presents many advantages as it deals with several issues of other transfer modes, generates a readable and editable format which provides efficiency and it is easy to implement.

When the hosting platform is successfully authenticated, the native machine prepares its agent to move. For that purpose, an agent named "BinSerial_Ag" is integrated, which takes in charge to serialize the instances of the mobile agent class (object), using Serializable and Externalizable interfaces along with a set of writing and reading methods on output and input streams. In this process, all attributes and variables of the mobile agent as well as its code are coded as a binary data flow. According to [15], the JAVA language provides tools (in JDK) that allow the binary serialization in transparent way and independently of the operating system.

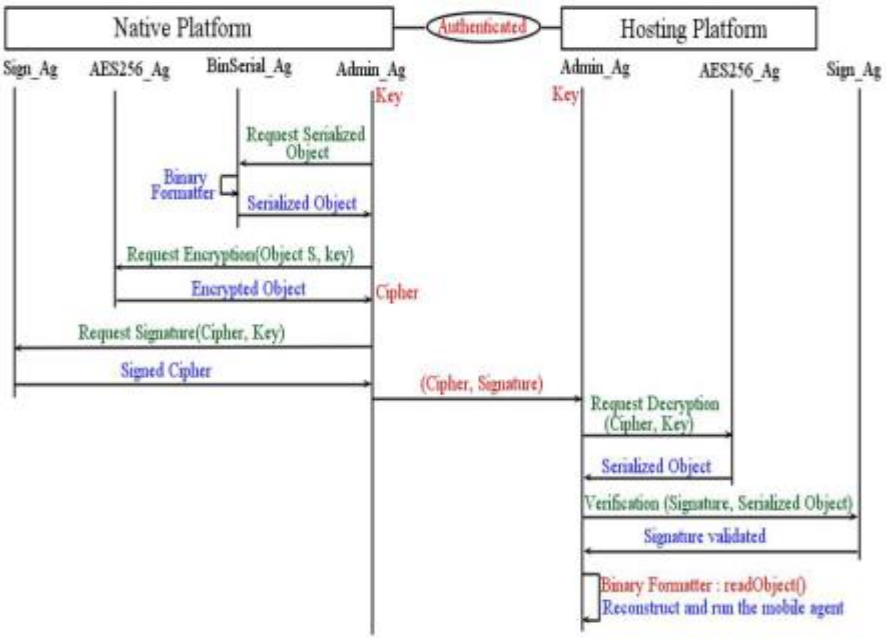

Figure 5. The Process of the Adopted Solution after Successful Authentication

Figure 5 shows where and when the process of binary serialization is introduced in the mobility of the agent. Before transferring the serialized instance of the mobile agent to the remote hosting platform, it is firstly encrypted using the session key generated in the authentication phase. Once arriving to the hosting platform, the mobile agent instance is decrypted and then deserialized using the BinaryFormatter of the JAVA serialization package. This aims to reconstruct the class of the serialized object on the intended platform. If the skeleton of the mobile agent class is not known by the API of the JAVA core, then the hosting platform may call the dynamic class loading services provided by JAVA.

3.3. Confidentiality of Data Exchanged

Data confidentiality consists on ensuring that information is kept secret and that only authorized persons have access. It is a much recommended property in elaboration of a security policy. Thus, maintaining the confidentiality of information circulating through the network is another issue, even it concerns the resources and services of the hosting platform or the code and data of the mobile agent.

The contribution adopted in this paper to fix this issue is based primarily on cryptographic algorithm. In general, cryptography is divided into two main categories: symmetric cryptography or secret key cryptography in which the communicating entities share the same key to encrypt and decrypt a message, while asymmetric cryptography or public key cryptography uses a pair of keys: a public key and a private key that are mathematically related by an encryption algorithm such that the message encrypted with a public key can be decrypted with the corresponding private key.

Each one of the categories mentioned has its advantages and its cons [16]. In one hand, public key cryptosystems are ranked to be more secure and effective as they are bound on the difficulty to solve complex problems such as the discrete logarithm. However, their complexity is translated with a slow processing and a huge consumption of resources, in addition to use lengthy keys that needs the allocation of a large space for storage. On the other hand, secret key cryptosystems are much faster in processing, as their computations are based essentially on permutations, substitutions and transformations. Also, they consume less memory and generate short keys that need to be shared through secure channels.

In the authentication phase, the ID-Based Key Agreement protocol allows us to share a session key of 256 bits in a secure manner, which solves the famous problem of securing the key exchange in symmetric cryptography. Hence, we think it very interesting to employ this category, especially for its high performance with the data compression and with devices having limited storage memory such a mobile agent. Thus, we choose the Advanced Encryption Standard (AES) [17]. It is a robust algorithm that introduces different key lengths (128, 196 or 256 bits) in different numbers of rounds (10, 12 or 14) depending on the key length. As we own a session key of 256 bits, then this matches very well with an AES algorithm of 128 bloc size, 256 bits key length and 14 rounds. In Figure 5, after that the mobile agent is serialized, it is sent by the “Admin_Ag” along with the session key to the “AES256_Ag”. This later encrypts the serialized object and returns it back to the “Admin_Ag” that takes in charge to lunch it migration process. Through encrypting the data exchanged between the authenticated platforms, we counter eavesdropping attacks and avoid that intruders know the real content of the message exchanged even it got it.

3.4. Schnorr's Signature for Integrity

The integrity is the principle made to ensure that the data are those believed to be, and verifying data integrity consists in checking whether the data have been altered during transmission (incidentally or intentionally) or not. It is a very valuable property to maintain confidence between the communicating parties.

To preserve the integrity of the exchanged data, each one of the communicating platforms signs the data that intends to send using Schnorr's signature [14] and associates it to the encrypted data, while the other one verifies if this signature matches with the decrypted data it has computed. Using the same mechanism shown in the last steps of Figure 4, where the hosting platform signs its identity along with ephemeral secret, and in contrast the native platform has to verify this signature to ensure authentication as well as to prove integrity.

When the hosting platform welcomes the mobile agent from the native platform, it receives in reality the encryption of the mobile agent serialized object along with the signature of the clear serialized object. Then the platform begins by decrypting the first part to obtain the plain format of the serialized object, which is latter signed using Schnorr signature and compared with the second part of the received data. If the both signatures match, then the integrity of the mobile agent received is proven. Then, the hosting platform can launch the execution of the mobile agent after proceeding to a reconstitution of its class with all attributes and functions, using Binary Formatter and the URLClassLoader if needed. Else, if the signatures are not corresponding, then the hosting platform terminates the mobile agent and sends a notification of integrity failure to the native platform.

The use of signature mechanism to support integrity of information communicated between two platforms, avoids several threats such as “Alteration Attacks” and “Pollution Attacks”. However, being able to prevent a platform from modifying an agent or at least detecting the changes occurred, still problems which didn't find yet a radical solution.

4. Evaluation and Results

In this section, we expose and discuss the results got through implementing the proposed solution. For that purpose we make use of JADE [9], the Java Agent Platform designed for multi-agent systems and with respect to specifications of FIPA [18] standard specified for software interoperability among agents and agent-based applications. First, we give a theoretical computation of the time spent to achieve all aspects of the solution. Then, we present the results of the testing performed to know the overhead provided through introducing security mechanisms.

During the trip of the agent from the native platform to the hosting platform, it performs many operations to ensure the different aspects of security needed. Let us consider T_{total} the total time spent for the solution. This period of time includes many processes represented by sub-durations:

$$T_{Total} = T_{Requests} + T_{ISAAC} + T_{IKA} + T_{Serialisation} + T_{Encryption} + T_{Signature} + T_{AgentMigration} + T_{Decryption} + T_{Verification} + T_{Deserialisation} \quad (1)$$

In order to simplify the computations, we will take into consideration some constraints. Knowing that the messages transporting requests are exchanged internally and their sizes didn't exceed a few characters, so $T_{Requests}$ can be considered as negligible. Also $T_{Encryption}$, is estimated to be approximatively equal to $T_{Decryption}$, as well as $T_{Serialisation}$ with $T_{Deserialisation}$. Then, the equation (1) of the total time spent for the solution execution becomes:

$$T_{Total} = T_{ISAAC} + T_{IKA} + 2T_{Serialisation} + 2T_{Encryption} + T_{Signature} + T_{AgentMigration} + T_{Verification} \quad (2)$$

The practical testing of the solution implementation is conducted using one machine as a Native Platform and four others considered as Hosting Platforms. All machines are heterogeneous with different operating systems (2 Windows, 2 Ubuntu, and 1 MacOs). The technical characteristics of the five machines are: Intel Core i7 processor at 2.7 GHz with 4 Go of RAM, connected through 100Mbps switched Ethernet network with WampServer [19]. For the creation, management, mobility and execution of the agents we adopt JADE Snapshot of version 4.3. For the IKA-Authentication protocol, SHA-2[20] and SHA-3[21] are used to generate a hash of 256 bits. Table 1 shows the timing results obtained during the trip of the mobile agent and the execution of different security processes.

Table 1. Time Performance of the Mobile Agent during a Round Trip

Number of Hosts	T_{ISAAC}	T_{IKA}	$T_{Serialisation}$	$T_{Encryption}$	$T_{Signature}$	$T_{AgentMigration}$	$T_{Verification}$
1	*2,9 ms*	*8,7 ms*	*59,5ms*	*14ms*	*12,7ms*	*179,4ms*	*2,7ms*
2	*5,6 ms*	*18,4ms*	*92,2ms*	*31,3ms*	*24 ms*	*343,2ms*	*5,2ms*
3	*8,5 ms*	*32,3ms*	*148,7ms*	*48,4ms*	*33,9ms*	*496,4ms*	*7,8ms*
4	*11,4 ms*	*51,1ms*	*182,6ms*	*82,8ms*	*45,6ms*	*682,7ms*	*10,2ms*

According to the equation (2) and referring to results obtained, the total time spent to execute our solution for one hosting platform is:

$$T_{Total} = 2{,}9 + 8{,}7 + (2 \times 59{,}5) + (2 \times 14) + 12{,}7 + 179{,}4 + 2{,}7 = 340{,}7ms$$

In the logic of calculus theory, the time that takes the agent to move over four hosting platforms, must be equal to four times the time of migrating the agent to one hosting platform. We think it interesting to see if our solution and the architecture chosen respect this theory settlings. Thus, to know the overall difference in our case, we calculate the total time to execute our solution for 4 hosting platforms:

$$T_{Total \times 4} = \frac{11{,}4 + 51{,}1 + (2 \times 182{,}6) + (2 \times 82{,}8) + 682{,}7 + 10{,}2}{4} = 333{,}05ms$$

Therefore, the overall difference is: $T_{Total \times 4} - T_{Total} = 333{,}05 - 340{,}7 = -7{,}65ms$ The negative value of the overall difference calculated shows that our solution and architecture not only respect the logic of the theory calculus, but also it scales with the change of environments and with the increase of visited hosts. This mainly due to the dynamic and flexible aspects of the agent behaviors, also the use of JAVA as programming language allows us to benefit from its tracing services that store traces of execution code in cash memory.

Finally, from above experiments we can extract the overhead of the security added for the mobility of the agent. This overhead is strongly related to the mechanisms employed in view of avoiding threats raising due to unavailability of authentication, integrity and confidentiality. Hence, the security overhead includes the cost of ISAAC+ random number generator, IKA-Authentication protocol, AES encryption and Schnorr signature. The computational value of this overhead is:

$$T_{Security} = T_{ISAAC} + T_{IKA} + 2T_{Encryption} + T_{Signature} + T_{Verification} = 55ms$$

This security overhead represents 16% of the total time spent to perform a mobility of the agent using the proposed solution. In practice, this value seems to be eligible and credible, and did not affect the flexibility of the mobility feature, which in fact gains in matter of protecting him from any external threats.

5. Conclusion

This paper discusses the security issues that a mobile agent may face while migrating over hosting platforms across the network. These issues are mainly related to the deficiencies of authentication, integrity, and confidentiality and availability aspects. Thus, we have elaborated a solution in which we attempt to address these security problems, using IKA protocol for authentication and key exchange, AES encryption to ensure confidentiality, Schnorr signature to preserve integrity and binary serialization to support easy and flexible mobility. This solution shows its effectiveness to prevent "Reflection Attacks", "Key Compromise Impersonation Attacks", "Eavesdropping Attacks", "Unauthorized Access" and "Alteration Attacks". Besides, the results of the solution timing performance indicate its scalability and efficiency.

As perspective, we will develop the other categories of threats in the systems based on mobile agents. Primarily, we would like to fix the security issues that a honest platform may encounter in hosting malicious mobile agents. Thereafter, we will be interested in conceiving a security model that combines solutions for protecting agent and platform in parallel.

References

[1] J. Ferber, "Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence", Editor Addison - Wesley, Reading, MA, **(1999)**.
[2] L. Zhou, A. S. Mohammed and D. Zhang, "Mobile personal information management agent: Supporting natural language interface and application integration", Information Processing and Management, vol. 1, no. 48, **(2012)**, pp. 23-31.
[3] M. Fasli, "On agent technology for e-commerce: trust, security and legal issues", The Knowledge Engineering Review, vol. 1, no. 22, **(2007)**, pp. 3-35.
[4] R. H. Glitho, O. Edgar and P. Samuel, "Mobile agents and their use for information retrieval: a brief overview and an elaborate case study", IEEE Network, vol. 1, no. 16, **(2002)**, pp. 34-41.
[5] D. Gavalas, G. E. Tsekouras and C. Anagnostopoulos, "A mobile agent platform for distributed network and systems management", Journal of Systems and Software, vol. 2, no. 82, **(2009)**, pp. 355- 371.
[6] F. Dignum, "Agents for games and simulations", Autonomous Agents and Multi-Agent Systems, springer, vol. 2, no. 24, **(2012)**, pp. 217-220.
[7] M. Metzger and G. Polakow, "A survey on applications of agent technology in industrial process control", IEEE Transactions on Industrial Informatics, vol. 4, no. 7, **(2011)**, pp. 570-581.
[8] B. Chen and H. Cheng, "A review of the applications of agent technology in traffic and transportation systems", IEEE Transactions on Intelligent Transport Systems, vol. 2, no. 11, **(2010)**, pp. 485-497.
[9] F. Bellifemine, A. Poggi and G. Rimassa, "JADE: A FIPA2000-compliant agent development environment", Editors J. Mller, E. Andre, S. Sen and C. Frasson (ed.), Proceedings of the 5th International Conference on Autonomous Agents, **(2001)** May 28-June 01, Montreal, Canada.
[10] P. Ahuja and V. Sharma, "A Review on Mobile Agent Security", International Journal of Recent Technology and Engineering (IJRTE), vol. 2, no. 1, **(2012)**, 2277–3878.
[11] R. Canetti and H. Krawczyk, "Universally Composable Notions of Key Exchange and Secure Channels", Editors, Advances in cryptology Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT), LNCS, no. 2332, **(2002)** April 28-May 02, Amsterdam, Netherlands.
[12] D. Fiore and R. Gennaro, "Making the Diffie-Hellman protocol identity-based", Editors, Topics in Cryptology-CT-RSA, Springer Berlin Heidelberg. Proceeding of the 10th Cryptographers' Track at the RSA Conference, **(2010)** March 1-5, San Francisco, CA, USA.
[13] J. Aumasson, "On the pseudo-random generator isaac", IACR Cryptology. ePrint Archive, **(2006)**.
[14] C. P. Schnorr, "Efficient signature generation by smart cards", Journal of Cryptology, vol. 3, no. 4, **(1991)**, pp. 161-174.

[15] C. J. Tauro, N. Ganesan, S. Mishra and A. Bhagwat, "Object Serialization: A Study of Techniques of Implementing Binary Serialization in C++, Java and. NET", International Journal of Computer Applications, vol. 6, no. 45, **(2012)**.
[16] R. Babu, G. Abraham and K. Borasia, "A Review On Securing Distributed Systems Using Symmetric Key Cryptography", International Journal of Advances in Science and Technology, vol. 4, no. 4, **(2012)**.
[17] T. Robertazzi, "Advanced Encryption Standard (AES)", In Basics of Computer Networking, Edited Springer New York, **(2012)**, pp. 73-77.
[18] "Foundation for Intelligent Physical Agents", FIPA ACL Message Structure Specification, **(2003)** August 01, http://www.fipa.org/specs/fipa00061/XC00061D.pdf.
[19] WampServer, http://www.wampserver.com.
[20] H. Gilbert and H. Handschuh, "Security Analysis of SHA-256 and Sisters", Editors, Selected Areas in Cryptography Proceedings of the 10th annual workshop, SAC, (**2003)** August 14-15, Ottawa, Canada.
[21] A. Jaffar and C. J. Martinez, "Detail Power Analysis of the SHA-3 Hashing Algorithm Candidates on Xilinx Spartan-3E", International Journal of Computer and Electrical Engineering, vol. 4, no. 5, **(2013)**, pp. 410-413.

