International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-8 Issue-12, October 2019

Mobile Agent Security Based on Mutual Authentication and Elliptic Curve Cryptography

Yousra Berguig, Jalal Laassiri, Sanae Hanaoui, Salah-ddine Krit

Abstract*: Mobile agent system is a satisfying solution for the implementation and maintenance of applications distributed over large-scale networks, this solution is very used in solving complex problems since they are autonomous, Intelligent, robust and fault-tolerant. Mobile agents have the capacity to migrate from one node to another all over the network allowing reduction in communication costs. Although they possess all these advantages, using them in distributed environment increases the threat to mobile agent security and during their mobility they can face different types of attacks such as of attacks like Replay attack, man-in-the-middle attack, Cookie theft attack, Offline password guessing attack, Stolen-verifier attack. In this paper we investigate the security of distributed mobile agent system. We propose a solution based on a secure Elliptic Curve Cryptography (ECC) protocol to ensure mutual authentication and protect the agent from different known attacks. The implementation of the proposed solution is obtained using Java Agent Development Framework (JADE). Also, Binary serialization is used to establish a flexible portability of the agent. Finally, we present security and performance analysis, for our solution to secure mobile agent in distributed systems.

Keywords: **Binary Serialization, Elliptic Curve Cryptography (ECC), JADE, Mobile Agent, Mutual Authentication, Security.**

I. INTRODUCTION

Mobile agent is a mobile object that moves from one host to another under the control of its own will to achieve tasks. It's an emerging technology that simplify the design, implementation and maintain of distributed systems. Recently, mobile agent technology becomes one of the active areas of research, it's widely used in several disciplines, like electronic commerce, industry, information retrieval, intrusion detection [4], and health care.

Revised Manuscript Received on October 30, 2019.
***** Correspondence Author
Yousra Berguig*, Informatics Systems and Optimization Laboratory, Ibn Tofail University, Faculty of Science, kenitra, Morocco. Email: yousra.berguig@gmail.com
Jalal Laassiri, Informatics Systems and Optimization Laboratory, Ibn Tofail University, Faculty of Science, kenitra, Morocco. Email: laassiri@uit.ac.ma
Sanae Hanaoui, Informatics Systems and Optimization Laboratory, Ibn Tofail University, Faculty of Science, kenitra, Morocco. Email: sanae.hanaoui@uit.ac.ma
Salah-ddine Krit, Informatics Systems and Optimization Laboratory, Polydisciplinary Faculty, Ouarzazate, Morocco. Email: salahddine.krit@gmail.com

With the evolution of intelligent and autonomous systems the mobile agent becomes a new way of communication over heterogonous network environment, with an important number of advantages. We notice that mobile agents reduce the network traffic, provide some effective means to overcome the network latency. Through their ability to operate asynchronously and autonomously of the process that created them, they help to construct more robust and fault-tolerant system, allow gather of information and accomplish tasks in an optimum way [10, 20, 6, 11]. However, we soon realized that behind all these qualities some deep serious security issues were hidden, that can be summed up in three main categories, Agent-to-Platform attack, Agent-to-Agent attack and Platform-to-Agent attack, (See Figure 1). Hence agents, hosts and data should be protected [5]. Some security researchers lean towards this problem, to secure mobile agent migration. Nonetheless There are a few security solutions that ensure mutual authentication and secure the agent against the four categories of risk mentioned above.

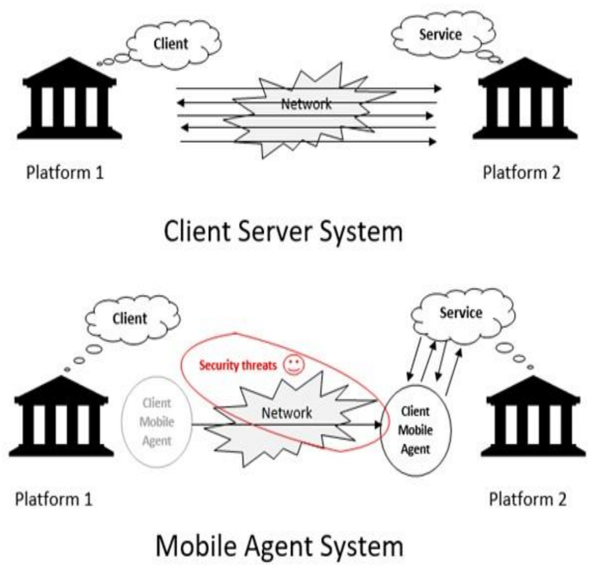

Fig. 1. Advantages and security problem in mobile agent system

Our main objective in this work is to elaborate a new secure scheme based on multi agent system and elliptic curve cryptography (ECC) [19]. Is a complex mathematical problem with a small key, which makes our solution autonomous, robust and powerful, as it's something that was not done before. The rest of the paper is organized as follows. In Section 2 we investigate the security problematic in mobile agents.

Subsequently, we mention some different possible threats that the agent can face during its mobility. Also, we present the preliminaries of elliptic curve cryptography. In Section 3, we enumerate some different countermeasure and related work that have been proposed and considered by researchers. In section 4, we propose and discuss our solution based on a secure Elliptic Curve Cryptography (ECC) protocol, to ensure mutual authentication. In section 5, we present our simulation results. Also, security and performance analysis will be shown in this section. Finally, section 6 concludes the paper.

II. MOBILE AGENT SECURITY AND ECC OVERVIEW

A. Overview on Mobile Agent Security

During the mobility, threats to mobile agent security are classified into four cases [1],[28]. The first one is agent-to-agent threats, when a malicious mobile agent tends to attack another agent; such as Masquerade, Denial of Service, Repudiation and Unauthorized access [7]. The second one is agent-to-platform threats, when a mobile agent becomes a threat to the destination platform. Here the mobile agent is malicious and may launch an attack over the Execution Platform. Attacks such as Masquerading, Denial of Service and Unauthorized Access come under this trait [7]. Platform to-agent threats, when the Execution Platform compromises the security of the Mobile Agent. This includes attacks such as Masquerading, Denial of Service, Eavesdropping and Alteration [7]. Finally, others-to-agent platform threats, this attack specifies all attacks which a Mobile Agent may suffer during its travel through the network or visiting a host. This may include masquerading, denial of service, unauthorized access and copy and-replay [7]. Then the mobile agent faces very serious security threats, since all its code, data, and state are exposed to the destination platform. We present below some of possible risks.

- **Leak out or modify mobile agent's code**

Malicious platform can read and remember the instructions that are going to be executed by the arrived mobile agent to infer the rest of the program. By this process the platform knows the strategy and purpose of mobile agents [13]. Sometimes the malicious platform has a complete picture of mobile agent's behavior and it might find out the physical address and then accesses its code memory to modify its code. It can even change code temporarily, execute it and finally resuming original code before the mobile agent leaves [8].

- **Leak out or modify mobile agent's data**

The malicious platform might get to know the original location of the data bit holed by the agent and then modify the data in accordance with the semantics of data [22], which might cause the leak of privacy or the loss of money that lead to severe consequences even if the data is not sensitive.

- **Leak out or modify mobile agent's execution flow**

The malicious platform can predict what will be set of instructions to be executed next and deduce the state of that mobile agent by knowing the mobile agent's physical location of program counter, mobile agent's code and data. Consequently, it can change the execution flow according to its will to achieve its goal [23]. It can even modify mobile agent's execution to deliberately execute agent's code in wrong way.

- **Denial of Service (DoS)**

This attack is one of the most dangerous attacks that might causes mobile agent to miss some good chances if it can finish its execution on that platform in time and travel to some other platform. It causes not to execute the mobile agent migration and put it in waiting list carrying delays [21].

- **Masquerading**

Here malicious platform pretends as if it is the platform on which mobile agent must migrate and finally becomes home platform where mobile agent returns. By this mechanism, it can get secrets of mobile agents by masquerading and even hurts the reputation of the original platform [29].

- **Leak out or Modify the interaction between a mobile agent and other parties**

Here the malicious platform eavesdrops on the interaction between a mobile agent and other parties (agent or platforms). This leads to extraction of secret information about mobile agent and third party. It can even alternate the contents of interaction and expose itself as part of interaction and direct the interaction to another unexpected third party. In this way, it might perform attacks on both mobile agent and third party.

B. Overview on elliptic curve (ECC)

ECC Definition

The ECC is an asymmetric algorithm it's an alternative of RSA which is the most common used for SSL certificates [2]. These two types of master keys share the same important property of having a key to encrypt and other to decrypt. However, ECC can offer the same level of encryption power for much shorter keys, providing better security while reducing computing requirements.

The shorter keys make ECC a very interesting and attractive option for devices with limited storage and processing power.

An elliptic curve E is curve given by a Weierstrass equation:

$$E: y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \quad (1)$$

We will consider in what follows an elliptic curve, is a curve that is drawn by the points that will solve the following equation:

$$E: (x, y) \,|y^2 \equiv x^3 + ax + b \quad with\ a, b \in K \quad (2)$$

a and b will have to fulfill the following condition $4a^3 + 27b^2 6= 0$, K can be in the following sets (R, Q, C, Z/pZ).

Symmetric	RSA	ECC
56	512	112
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

Fig. 2. ECC vs. RSA key size

Proposition

Let E be an elliptic curve defined on a field K, and two points $P, Q \in E(K)$, L the line connecting *P to Q* (the tangent to *E* if $P = Q$) and *R* the third intersection point of *L* on *E*. Let L be the vertical line passing through R. We define $P + Q \in E(K)$ as the second point of intersection of *L* with *E*. With the law of composition, $(E(K), +)$ is an abelian group whose neutral element is the point to infinity (O).

- **Point addition**: With 2 distinct points, P and Q, the addition is defined as the negation of the point resulting from the intersection of the curve, E, and the line defined by the points P and Q, giving the point, R.

$$\begin{aligned} &P + Q = R \rightarrow (x_p, y_p) + (x_q, y_q) = (x_r, y_r) \\ &x_r = \lambda^2 - (x_q + y_q) \\ &y_r = \lambda \times (x_p - x_r) - y_p \quad with \quad \lambda = \frac{(y_p - y_q)}{(x_p - x_q)} \end{aligned} \quad (3)$$

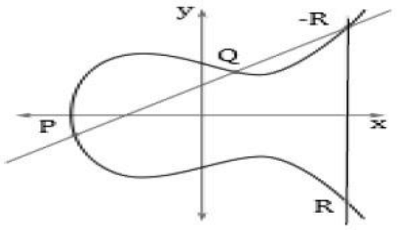

Fig. 3. Point addition

- **Point doubling:** When the points *P* and *Q* are coincident, the addition is similar, except that there is no straight line defined by *P* and *Q*, so the operation is closed using the limit case, the tangent to the curve *E*, to *P* and *Q*. This is calculated as above but with:

$$\lambda = \frac{(3x_p^2 + a)}{2y_p} \quad (4)$$

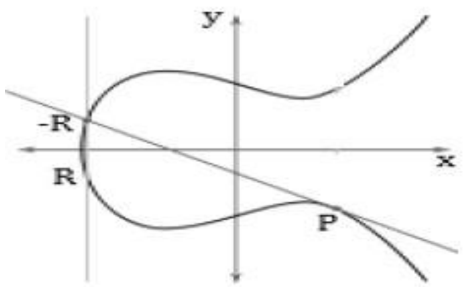

Fig. 4. Point doubling

- **Vertical point:** The straight line joining any point *P* and its symmetrical relative to the horizontal axis, noted $-P$, is a vertical line, the third point of intersection with the curve is the point at infinity (which is its own symmetrical with respect to the abscissa axis) hence $P + (-P) = 0$.

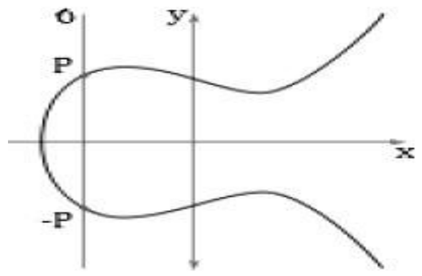

Fig. 5. Vertical point

- **Double-and-add**: The simplest method is the double-and-add method, similar to multiply-and-square in modular exponentiation. The algorithm works as follows: To compute DP, start with the binary representation for

$$d = d_0 + 2d_1 + 2^2 d_2 + \ldots + 2^m d_m \; with \; [d_0 \ldots d_m] \in [0,1] \quad (5)$$

III. RELATED WORKS

A. Mobile agent Security solutions

Nowadays Mobile-Agents are widely used in different distributed systems, smart devices and connected objects. Nevertheless, the security issue still represents significant constraints and the mobility of these entities needs to be secured. The field of Mobile-Agents security has many diverse research, approaches and ideas. Esfandi & Rahimabadi [9] proposed a multi agent-multi key approach where the encrypted private key and the message are broken into different parts carrying by different agents which makes it difficult for malicious entities to mine the private key for message encryption. To improve security, they used Advanced Encryption Standard (AES) for message encryption. X. Hong [14] presented threshold proxy signature protocol that uses a proxy signer to sign a digital signature. The proxy signing process uses RSA algorithm and it is shared using Lagrange Formula. Sabir et al. [26] proposed a new Authentication and load balancing scheme base on Json to answer the communication security problem in Multi-Agent environment. The presented model based on a Multi-Agent system takes advantage of JWT's stateless functionality to ensure the integrity of the exchanged messages between agents, the authentication of the agents, and non-repudiation, based on the asymmetric cryptographic technology. Sampathkumar [17] proposed solutions based on public key authentication technique and cryptography to address some of security problems in Mobile agent technology, they present also an experimental application by using RSA algorithm to encryption and decryption. Idrissi et al. [16] introduce an approach based on Identity-Based Key Agreement Protocol to get a session key and to ensure authentication, an Advanced Standard Encryption (AES) for the confidentiality of data exchanged, as well as a Binary Serialization. Unfortunately, we noticed that in literature the most cryptographic approaches and solutions for Mobile-Agent security are based on the classical Cryptosystem like AES, RSA...

In our work we choose to use the elliptic curve for many reasons, one of the main reasons is the key size that is very small compared to other asymmetric cryptosystems, it requires less computational power, communication, bandwidth, and memory. Also, its complexity that is very difficult to calculate. For this we used Kumari et al. [5] algorithm that propose a mutual authentication to secure the communication between IoT devices and the Cloud. The proposed scheme is invincible to various attacks and they use HTTP cookies which make it very interesting.

B. Elliptic curve cryptography for system security

X. Huang et al. [15] present an approach for protecting a system from man-in-middle attacks based on hidden generator point with elliptic curve cryptography (ECC), they designed a hidden generator point that offer a good protection from (MITM) attack and opted for multiagent system implementation. J R. Shaikh et al. [27] present an analysis of various types of curves recommended by different standards, by performing two ECC algorithms - ECDH and ECDSA. and offer a comparative table of selected curves that is arranged according to the computation time taken by each curve to perform various operations when used for the ECC algorithms. N. Mehibel & M. Hamadouche [24], propose a new approach of elliptic curve to secure Diffie-Hellman key transport in public channel. D. Pritam Shah & P. Gajkumar Shah [25] propose a secured protocol based on elliptic curve for communication between IoT devices and server. In the proposed Elliptical Curve Internet of Things (ECIOT) protocol, the IoT device will establish secret session key with server by using Diffie-Hellman protocol based on NIST p-192 prime curve. The subsequent communication will be carried out with symmetric key cipher Ex-OR by using ECIOT derived key. Keerthi K & B.Surendiran [18] introduce a new mapping technique for encoding the message into affine points on the elliptic curve. Mapping technique convert the plain text into ASCII values and then convert this into hexadecimal. The converted Hex values are grouped together to form the x and y coordinates. The converted values are encrypted in reverse order to prevent security attacks.

IV. ELLIPTIC CURVE CRYPTOGRAPHY FOR MOBILE-AGENT SECURITY

A. Model of our solution

Recently mobile agents are widely used in distributed systems in order to communicate, execute tasks and exchange information and sensible data between the interacting entities, nonetheless it is noted that mobile agent security has become progressively pronounced. This part gives a description of the proposed solution based on S. Kumari et al. ECC algorithm [19] to keep the mobile agent platform and the agent itself secure against different threats.

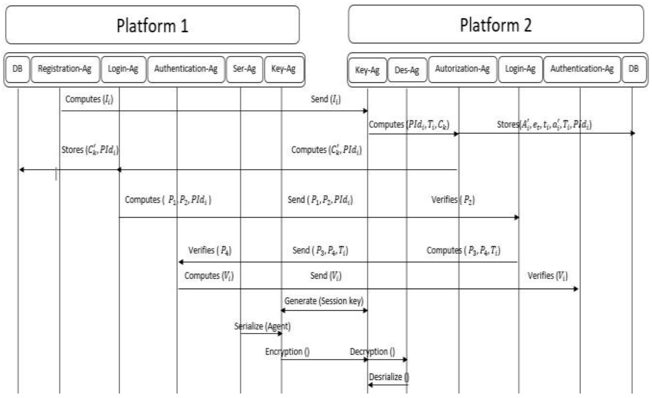

Fig. 6. Proposed solution Model

Initialization

First, RC chooses an elliptic curve (EC) equation $y^2 = x^3 + ax + b$ over Z_p where $Z_p(p > 2^{160})$ is the finite field group. Then selects two elements $a, b \in Z_p$, where a and b satisfy the condition $4a^3 + 27b^2 (mod\ p) \neq 0$. The EC base point is G that has a prime order of n ($n > 2^{160}$). Let O be the point at infinity satisfying the equation $n \cdot G = O \cdot a$, random nonce X_{RC} is selected by RC as its secret key.

Registration

Step 1. The Registration-Ag is the agent responsible of the registration with RC. The agent computes $I_i = h(Id_i \| Pw_i)$ and sends I_i to the Key-Ag2 through a secure communication channel.

Step 2. After receiving the registration request. The Key-Ag2 generates a random number r_s and computes $PId_i = h(r_s \| Id_{RC} \| I_i) \oplus Id_{RC}$ and stores PIdi. Then, computes the cookie $C_k = h(r_s \| X_{RC} \| E_t \| PId_i)$ and $T_i = r_s \oplus h(X_{RC} \| PId_i)$ and send those parameters to Authorization-Ag.

Step 3. Authorization-Ag computes some other security parameters $C_k' = C_k \cdot G$, $A_i = h(r_S \oplus h(X_{RC} \| PId_i) \oplus I_i \oplus C'_k)$, $A'_i = A_i \cdot G$, $t_i = T_i \oplus X_{RC}$ $e_t = E_t \oplus X_{RC}$ and $a'_i = A'_i \oplus X_{RC}$. Then C_k' and PId_i are sent to Registration-Ag that stores theme in the database.

Login and authentication phase

Step 1. The Login-Ag1 selects random nonce r_1 before each login. Then compute the ECC point $P_1 = r_1 \cdot G$ and $P_2 = h\ (r_1 \cdot C_k')$, and, stores P_1 in its memory and sends the login request P_1 ,P_2, PId_i to Login-Ag2.

Step 2. After receiving the login request, Login-Ag2 obtains P_1, P_2 and PIdi and computes $r_s = T_i \oplus h(X_{RC} \| PId_i)$, then computes the cookie information $C_k = h\ (r_s \| X_{RC} \| E_t \| PId_i)$ and $P{*}_2 = h\ (P_1 \cdot C_k)$.

Next, it verifies $P{*}_2$ (? =) P_2. If it's true then Login-Ag2 selects a random nonce r_2, computes the ECC point $P_3 = r_2 \cdot G$; $P_4 = r_2 \cdot A_i'$ and sends P_3, P_4, T_i to Authentication-Ag1. Differently login-Ag2 rejects the login request of loginAg1.

Step 3. After receiving P_3, P_4, T_i Authentication-Ag1 compute $A_i = h\ (T_i \oplus I_i \oplus C_k')$ and the ECC point $P{*}_4 = P_3 \cdot A_i$ After that, it verifies $P{*}_4$(? =) P_4 to authenticate the platform 2, if it's true then Authentication-Ag1 authenticate platform 2 and computes the session key $SK = r_1 \cdot P_3 = r_1 \cdot r_2 \cdot G$ and sends $V_i = h((r_1 \cdot C'_k) \| SK)$ to Authentication-Ag2.

Retrieval Number: L34381081219/2019©BEIESP
DOI: 10.35940/ijitee.L3438.1081219
Journal Website: www.ijitee.org

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

Otherwise rejects the message P_3, P_4, T_i of the platform 2 and resume the login.

Step 4. Authentication-Ag2, compute the session key $SK* = r_2 \cdot P_1 = r_2 \cdot r_1 \cdot G$ and $V*_i = h((P_1 \cdot C_k) \| SK*)$. Then, verifies $V*_i$ (? =) V_i to authenticate the sender platform SD. If it's true then Authentication-Ag2 authenticates platform1, Differently, V_i is rejected.

Step 5. The key-Agents computes the session keys, $SK = r_1 \cdot P_3 = r_1 \cdot r_2 \cdot G = r_2 \cdot P_1 = SK*$.

Serialization and encryption

Step 1. To make mobile agent transportability easy and insistent we use the binary serialization mechanism that generates a readable and editable format. When the platform 2 is successfully authenticated, the platform 1 prepare its mobile agent to move. For this the Serialization-Ag is created to serialize the agent and sends the binary data flow to the key-Ag.

Step 2. After receiving the serialized agent Key-Ag1 XORed the binary data flow with the session key $SK = r_1 \cdot P_3$ and send the result to the keyAg2.

Step 3. The Key-Ag2 decrypt the message and send it to the DeserializationAg that deserialize the agent and run it.

List of notations

The table below shows the notation used in this paper

Notation	Description
SD	The sender platform
Id_i	Identity of Sd
Pw_i	Password of Sd
RC	The receiver platform
Id_{RC}	Identity of RC
X_{RC}	Secret key of RC based on ECC
Zp	Finite field group
P	Large prime number of the order $>2^{160}$
r1, r2	Random numbers generated for ECC parameters
r_s	Random number generated by RC
G	Generator point of a large order n
C_k	Cookie information
E_t	Expiration time of the cookie
$h(.)$	Cryptographic one-way hash function
$\oplus$	Bitwise XORed
$\|\|$	Concatenation

Fig. 7. Table of notation used

B. Algorithm presentation

In this part, we detail the different steps of the algorithm used to guarantee mutual authentication.

Algorithm 1: Initialization

Input: E_p: The elliptic curve equation, X_{RC}: Reception platform secret key Output: P_S: Reception platform public key, G: Generator point.

1. $E_p \leftarrow E: y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$
2. Choose the generator point G.
3. $P_s \leftarrow X_{RC} \cdot G$

Algorithm 2: Registration

Input: ID_i: Identity of the sender platform (SD), Pw_i: password of SD.

Id_{RC}: Identity of reception platform (RC) X_{RC}: secret key of RC

E_t: Expiration time of the cookie PId_i: pseudo Identity

C_k': The cookie information and a set of parameters T_i, A_i, t_i, a'_i, e_t.

Output: I_i: The hashed identity of SD, G: Generator point.

1. Computes $I_i \leftarrow h(ID_i \| Pw_i)$
2. Selecta random number r_s.
3. Computes $PId_i \leftarrow h(r_s \| Id_{RC} \| I_i) \oplus Id_{RC}$
4. Computes $C_k \leftarrow h(r_s \| X_{RC} \| E_t \| PId_i)$
5. Computes $C_k' \leftarrow C_k \cdot G$
6. Computes $T_i \leftarrow r_s \oplus h(X_{RC} \| PId_i)$
7. Computes $A_i \leftarrow h(r_s \oplus h(X_{RC} \| PId_i) \oplus I_i \oplus C'_k)$
8. Computes $A'_i \leftarrow A_i \cdot G$
9. Computes $t_i \leftarrow T_i \oplus X_{RC}$
10. Computes $a'_i \leftarrow A'_i \oplus X_{RC}$
11. Computes $e_t = E_t \oplus X_{RC}$
12. Stores t_i, a'_i, e_t and PId_i in the data base
13. Sends PId_i and C_k' to the sender platform SD
14. The sender platform stores PId_i and C_k' in its data base

Algorithm 3: Login and Authentication

Input: r_1, r_2: Selected random number, X_{RC}: secret key of RC E_t: Expiration time of the cookie, G: Generator point

Output: P_1,P_2,P_3,P_4: ECC points, C_k: The cookie information, Sk: Session key and a set of parameters r_s, V_i, $V*_i$

1. Select a random number r_1
2. Computes $P_1 \leftarrow r_1 \cdot G$.
3. Computes $P_2 \leftarrow h(r_1 \cdot C_k')$
4. Sends P_1, P_2, PId_i to the receiver platform
5. Computes $r_s \leftarrow T_i \oplus h(X_{RC} \| PId_i)$
6. Computes $C_k \leftarrow h(r_s \| X_{RC} \| E_t \| PId_i)$
7. Computes $P*_2 \leftarrow h(P_1 \cdot C_k)$
8. Verifies $P*_2$ (? =) P_2
9. Computes $P_3 \leftarrow r_2 \cdot G$ and $P_4 \leftarrow r_2 \cdot A'_i$
10. Sends P_3, P_4 and T_i to the sender platform SD
11. SD Computes $A_i \leftarrow h(T_i \oplus I_i \oplus C'_k)$
12. SD computes $P*_4 = P_3 \cdot A_i$
13. Verifies $P*_4$ (? =) P_4
14. Computes $SK \leftarrow r_1 \cdot P_3$.
15. Computes $V_i \leftarrow h((r_1 \cdot C'_k) \| SK)$
16. Sends V_i to RC
17. Computes $SK* \leftarrow r_2 \cdot P_1 \leftarrow r_2 \cdot r_1 \cdot G$.
18. Computes $V*_i \leftarrow h((P_1 \cdot C_k) \| SK*)$.
19. Verifies $V*_i$ (? =) V_i
20. Generates the session key

$SK \leftarrow r_1 \cdot P_3 \leftarrow SK* \leftarrow r_2 \cdot P_1 \leftarrow r_2 \cdot r_1 \cdot G$.

Algorithm 4: Serialization and encryption

Input: M_A: The instance of Mobile agent, SK: Session key, MA_S: Serialized mobile agent Output: MA_{DS}: Deserilized mobile agent, MA_{cf}: encrypted Mobile agent.
1.$MA_S \leftarrow$ Serialize M_A
2.$SK \leftarrow r_1 \cdot P_3$
3.Computes $MA_{cf} \leftarrow MA_S \oplus SK$
4.Decrypt and desterilized MA_{cf}

V. IMPLEMENTATION AND RESULTS

In this section we present the results got through implementing the proposed solution. For that we used the test platform JADE (Java Agent development framework) [20, 30], a platform that meets the standard FIPA [3] specified for agents and uses the Java language it's the most widespread and it's offer more services inter-operability for facilitating the mobility of workers.
The practical tests of the implementation are carried out in a Machine which contains two containers that will represent the source machine and the destination machine. Table 1 shows the characteristics of machine used in this experiment.
Our solution is implemented using a set of agents as we mentioned above.

Table- I: The platforms characteristics

Platform type	Container Name	RAM (MB)	Processor	OS
PC	Container-1 Container-2	8GO	Intel Core i5 2.3 GHz	Windows 10

First, we have Registration-Ag which is responsible for the calculation and sending I_i, we find later Key-Agent1 which computes a set of parameters (PID_i, T_i, C_k) and store them in the database of the platform. Then we have Authorization-Ag which computes ($A'_i, C'_k, a'_i, e_t, t_i$). We have also, Log-Ag that computes (P_1,P_2) and sends it to Log-Ag1.This agent verifies (P_2,$P*_2$), calculates (P_3,P_4) and send it to Auth-Ag, that verifies (P_4,$P*_4$), calculates V_i and sends it to Auth-Ag1. Auth-Ag1 is responsible for checking the authentication. Also, we have Key-Ag and Key-Ag1 their roles are the keys generation, encryption and decryption. Finally, we have ser-Ag and des-Ag for serialization and deserialization of the agent. In figure "6" we present all the agents used in our implementation and interactions between them.
In order to achieve the implementation of our solution, we used a main class "Scalar-Multiply" which generates the EC-point G and allows us to get scalar multiples. We used the curve P–192 which is given by the following parameters of Base point G: [31]
X_G = 602046282375688656758213480587526111916\
98976636884684818
Y_G = 174050332293622031404857552280219410364\
023488927386650641
For the creation, management, mobility and execution of the agents we. adopt JADE of version 4.5. SHA-256 [32] is used to generate hash values. our solution is based on four parts, Registration, Login, Authentication and Serialization. In the following we present the result of each step of our solution.

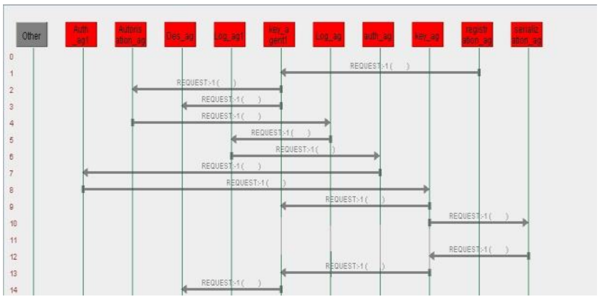

Fig. 8. Model generated by our implementation

Before presenting the results of the tests to know the overheads generated by the introduction of security mechanisms, we propose a theoretical calculation of the time required to carry out all aspects of the solution. During migration Mobile-Agent performs various operations to ensure the different aspects of security needed. We consider T_{total} the total time for the solution. T_{total} is composed of sub-duration that represent every step of the solution.

$$Ttotal = Trequests + Tregister + Tlog + Tauthentication + Tserialization + Tencryption + Tmigration + Tdecryption + Tdeserialization. \quad (6)$$

```
hey im the  Auth_ag1@192.168.1.11:1099/JADE agent
hey im the  Autorisation_ag@192.168.1.11:1099/JADE agent
hey im the  key_agent1@192.168.1.11:1099/JADE agent
************************Registration phase************************
**********Key Agent1**********
Message recu 1 -> df9039bc6b825dea47522ebbb155a91c159f83c83ed0e38cc4a27b3d8ea7af04
////////PIDi D79A574DF0F3D1C9325DD1D0577AD74DF1F3D3FA3259C85A4F6854C751955750174DCAB65BFA17D37A1653E1D54FA95B/////////
//////////TI CC//////////
//////////CK 675bb9f4a08afbf90312bf42f9c88c59ab7c1d9cc00e6cbe302ef7dbefabe75d//////////
message envoye a Autorisation_Ag
**********Autorisatio agent**********
Message recu 2 -> D79A574DF0F3D1C9325DD1D0577AD74DF1F3D3FA3259C85A4F6854C751955750174DCAB65BFA17D37A1653E1D54FA95B 675bb9f4a08afbf90312bf42f9c
////ck.G_x 4258469966304615226865061592105624016655262434968164556703
////////CK'4258469966304615226865061592105624016655262434968164556703 5773462201182954068585074056863887115494668959301882191433////////
////////Iidf9039bc6b825dea47522ebbb155a91c159f83c83ed0e38cc4a27b3d8ea7af04////////
////////Ai 4D////////
////Ai.G_x 3698977842746993817446189245208741847414075700581440652742
////////Ai' 3698977842746993817446189245208741847414075700581440652742 4267012988908307679832337725261123626684097361508249132856////////
message envoye a Log_Ag
```

Fig. 9. Result of the authorization phase

```
**********Login Phase**********
**********Login_agent_1**********
Message recu 5 -> 3176317450453705650283775811228493626776489433309636475023 4460189377466938476679380385498011517961211807501706
///random1///13
**********Ti_FILE**********CC
**********pidi_FILE**********D79A574DF0F3D1C9325DD1D0577AD74DF1F3D3FA3259C85A4F6854C751955750174DCAB65BFA17D37A1653E1D54FA95B
**********CK_FILE**********675bb9f4a08afbf90312bf42f9c88c59ab7c1d9cc00e6cbe302ef7dbefabe75d
**********Apx_FILE**********3698977842746993817446189245208741847414075700581440652742
**********Apy_FILE**********4267012988908307679832337725261123626684097361508249132856
////////rs1 [B@71547010/////////
////////rs2 [B@c0a1991/////////
////////rs3 [B@9491160/////////
////////rs A974D340CBD3153207F5347E40022C21D98E9A49510013D859457B1B69B95AEA/////////
////////ck 0cb40f6e6db6883a8d662fed76352bf357a13505cc31e14c90bd800270eab190/////////
////////OKK 46750289905179555494220781901621200449603706611806924457736380706802935523165/////////
////p22 4778091183731822747809733668495059901318857112616336195858
p22x : 4778091183731822747809733668495059901318857112616336195858
////////P2 4778091183731822747809733668495059901318857112616336195858 2009820340295151011960794752314141455652435103568189833321/
////////P2H 258e7f027b81b2396e44176794bf1562fdb7fb5aa06fe9aa342fafc7a7139949/////////
/////////P2*=P2/////////
```

Fig. 10. Result of the Login phase

```
////////R216 13/////////
////sk* 2823183308804106225556300899794101437277395293213606406558
skpx : 2823183308804106225556300899794101437277395293213606406558
////////SK* 2823183308804106225556300899794101437277395293213606406558 5991116009891131797391997193798939663353898992415047467865////
////////CK16 46750289905179555494220781901621200449603706611806924457736380706802935523165/////////
////RC* 4778091183731822747809733668495059901318857112616336195858
RC*x : 4778091183731822747809733668495059901318857112616336195858
////////RC* 4778091183731822747809733668495059901318857112616336195858 2009820340295151011960794752314141455652435103568189833321////
////////V 47780911837318227478097336684950599013188571126163361958582823183308804106225556300899794101437277395293213606406558///////
////////Vi* 10452e7bcd121892eef12b6609efbf0328e7e94686c8666be0b1a30ef7e4a9c1////////
/////////!!!!Vi*=Vi!!!!/////////
*******************Successful Authetication**********************
```

Fig. 11. Result of the authentication phase

```
*******serialization and encryption of Mobile_Agent*******
mob@192.168.1.11:1099/JADE
Serialized Agent[-84, -19, 0, 5, 115, 114, 0, 16, 109, 115, 103, 46, 77, 111, 98, 105, 108, 101, 95, 65, 103, 101, 110, 116, 72, 88, -74, 8, -
cipher message: 7958000D4F134BA64B4D2575AB711C7969B14720A52118776114110125
send message: 7958000D4F134BA64B4D2575AB711C7969B14720A52118776114110125
```

Fig. 12. Result of the Serialization and encryption phase

```
******Decryption and deserialization of Mobile-Agent******
message received 7958000D4F134BA64B4D2575AB711C7969B14720A52118776114110125
decrypted message: [-84, -19, 0, 5, 115, 114, 0, 16, 109, 115, 103, 46, 77, 111, 98, 105, 108, 101, 95, 65, 103, 101, 110, 116, 72, 88, -74, 8
deserilized message mob@192.168.1.11:1099/JADE
```

Fig. 13. Result of the Deserialization and decryption phase

Since the requests between agents are internal with a small size. Also, in our simulation case the agent migrates internally from one container to another so we can consider $T_{requests}$ and $T_{register}$ as negligible. Approximatively $T_{serialization}$ is equal to $T_{deserialization}$, as well as $T_{decryption}$ is equal to $T_{encryption}$. Therefore, the equation (3) of the solution total time becomes:

$$T_{total} = T_{register} + T_{log} + T_{authentication} + 2T_{serialization} + 2T_{encryption}. \quad (7)$$

Finally, conforming to the equation (7) the total time spent to execute our solution is:

$T_{total} = 135 + 81 + 84 + 2 \times 140 + 2 \times 5 = 590ms$

Table- II. Timing results obtained in the execution of every step

$T_{register}$ (ms)	T_{log} (ms)	$T_{authentication}$ (ms)	$2T_{serialization}$ (ms)	$2T_{encryption}$ (ms)	T_{total} (ms)
135	81	84	280	10	590

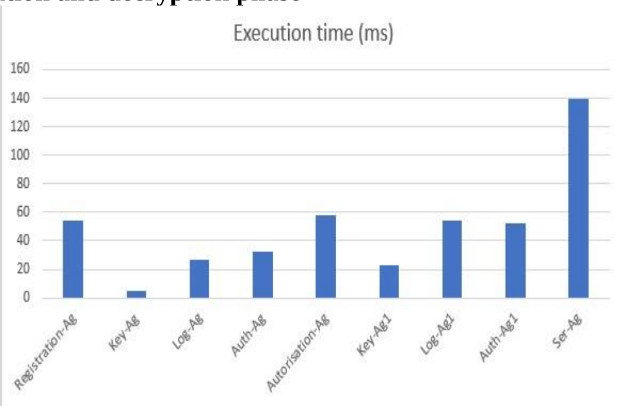

Fig. 14. Timing results obtained in the execution of every agent used in the simulation.

Despite that we used asymmetric cryptography that takes a little longer time during its execution. The overhead of the security added for the mobility of the agent is 310 ms which appears admissible, credible and not compromising the performances of our agent platform.

Fig. 15. Timing results obtained in the execution of different security processes

Security Analysis

Our solution is based on the ECC algorithm of S. Kumar et al.[19] achieves a variety of security objectives and protects the agent from various types of attacks such as Replay attack, man-in-the-middle attack, Cookie theft attack, Offline password guessing attack, Stolen-verifier attack, Impersonation attack and more. It guarantees mutual authentication, confidentiality and device anonymity. In this part, we will compare our solution with two others based also on a set of cryptographic algorithms. First H. Idrissi et al. [16] proposed a very inspiring and interesting solution based on Identity-Based Key Agreement Protocol and Advanced Standard Encryption (AES) which is a symmetric-Key algorithm. The execution time of this solution is 55ms which is less than our execution time, this is justified by the use of symmetric cryptography which is an old technique uses a single key that needs to be shared among the parts who need to receive the message.

The problem in this type of encryption is the use of a single key for encryption and decryption, if the key is stolen while transmitting data between sender and receiver it is very easy to decrypt the message. This work does not guarantee the availability of the agent .In the work "On the Security Communication and Migration in Mobile Agent Systems" [12], authors proposed an interesting solution based on a set of security mechanisms such as asymmetric-Key algorithm Rivest–Shamir–Adleman (RSA), Schnorr signature and an ECC elliptic curve algorithm to secure the agent during its mobility. However, the time execution for the hole solution is 5000 ms which is very big compared to our execution time in the current paper. This could be explained by the different asymmetric algorithms and signature used to guarantee the authentication, confidentiality, integrity and availability of the agent. Which require an important execution time. Nonetheless in our solution we guarantee all this security requirements by using a single ECC algorithm in 310 ms.

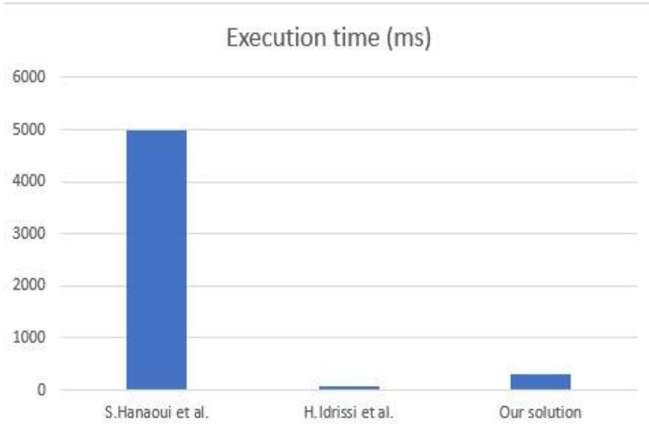

Fig. 16. Evaluation of the execution time of mobile-agent security solution for our Approach and two other approaches

VI. CONCLUSION

This work deals with the problem of mobile agent security. The proposed solution is a novel contribution based on elliptic curve cryptography to secure mobile agent platforms against unauthorized access attacks. and binary serialization to guarantee an easy and flexible mobility. We offer a solution that protect the agent from different types of attacks like Replay attack, man-in-the-middle attack, Cookie theft attack, Offline password guessing attack, Stolen-verifier attack. We implemented our solution using JAVA programming language and JADE (Java Agent Development Framework) which allows us to simulate our solution between two platforms using a multi-agent system. The results of the solution timing performance show its efficiency and adaptability. In the following work we will extend our scheme by using our own asymmetric algorithm for mobile agent security.

REFERENCES

1. J.J. Adri Jovin, M. Marikkannan. A Review on Attacks and Security Approaches in Mobile Agent Technology. Australian Journal of Basic and Applied Sciences, 2016.
2. Alfred, J. Menezes, Paul C. Van Oorschot, Scott A. Vanstone. Handbook of applied cryptography. CRC press, 1996.
3. F. Bellifmine, A. Poggi and G. Rimassa. JADE: a FIPA2000 compliant agent development environment. 216-217.ACM, 2001.
4. Y. Berguig, J.Lassiri, S. Hanaoui. DDoS Detection based on Mobile Agent and Na¨ıve Bayes Filter. International Symposium on Advanced Electrical and Communication Technologies (ISAECT), 2018.
5. P. Braun, W. Rossak. Mobile agent security: Basic concepts, mobility models and the tracytoolkit. Morgan Kaufmann/Elsevier anddpunkt. verlag, USA, 2005.
6. J. Cao, S. K. Das. Mobile agents in networking and distributed computing. (Vol. 3), Jhon Wiley & Sons, 2012.
7. G. Carl, G. Kesidis, RR. Brooks, S. Rai. Denial-of-service attack detection techniques Trans intern Comput 10(1):82–89, 2006.
8. P. Dadhich, D. K. Dutta, D. M. C. Govil. Security Issues in Mobile Agents. Int. J. Comput. Appl., vol. 11, no. 4, pp. 1–7, 2010.
9. A. Esfandi, A. Rahimabadi. Mobile Agent Security in Multi agent Environments Using a Multi agent-Multi key Approach, 2nd IEEE International Conference on Computer Science and Information Technology, 2009.
10. J. Ferber. Les syst`emes multi-agents. Versune intelligence collective. Inter Edition, Paris, 1995.
11. B. B. Gupta, Omkar P. Badve. Taxonomy of DoS and DDoS attacks and desirable defense mechanism in a Cloud computing environment. Neural Comput& Applic, 2017.

12. S. Hanaoui, Y. Berguig, J. Laassiri. On the Security Communication and Migration in Mobile Agent Systems. International Conference on Advanced Intelligent Systems for Sustainable Development. pp 302-313, 2019.
13. Y. Hedin, E. Moradian. Security in Multi-Agent Systems. Procedia Comput. Sci., vol. 60, no. 1, pp. 1604–1612, Jan. 2015.
14. X. Hong. Efficient threshold proxy signature protocol for mobile agents, Information Sciences 179, 4243–4248, 2009.
15. X. Huang, P. Gajkumar Shah, and D. Sharma. Multi-Agent System Protecting from Attacking with Elliptic Curve Cryptography, Advances in Intel. Decision Technologies, SIST 4, pp. 123–131, 2010.
16. H. Idrissi, E. Souidi and A. Revel. Mobile Agent Security Using IDBased Agreement Protocol and Binary Serialization, International Journal of Security and Its Applications Vol. 9, No. 5 , pp. 19-30, 2015.
17. A. Kannammal, N. Ch.S.N. Iyengar. A Framework for Mobile Agent Security in Distributed Agent-Based E-Business Systems, International Journal of Business and Information, Volume 3, Number 1, June 2008.
18. K. Keerthi, Dr. B. Surendiran. Elliptic Curve Cryptography for Secured Text Encryption. International Conference on circuits Power and Computing Technologies [ICCPCT], 2017.
19. S. Kumari, M. Karuppiah, A. Kumar Da, X. Li, F. Wu, N. Kumar. A secure authentication scheme based on elliptic curve cryptography for IoT and cloud servers. The Journal of Supercomputing, Volume 74, Issue 12, pp 6428–6453, 2017.
20. D.B. Lange, M. Oshima. Seven good reasons for mobile agents. Commun. ACM, ACM, vol. 42, 1999.
21. F. Lau, S. H. S. S. H. Rubin, M. H. Smith, L. Trajkovic. Distributed denial of service attacks. Syst. Man, Cybern. IEEE Int. Conf., vol. vol.3, no. SANS Security 401, pp. 2275–2280, 2000.
22. C. M.-F. for S. M. Code and undefined, Detecting attacks on mobile agents. pdfs.semanticscholar.org, 1997.
23. M. W. Madsen. An introduction to multi-agent statistics. CEUR Workshop Proc. vol. 1208, pp. 16–20, 2014.
24. N. Mehibel, M. Hamadouche. A New Approach of Elliptic Curve DiffieHellman Key Exchange, The 5th International Conference on Electrical Engineering, 2017.
25. D. Pritam Shah, P. Gajkumar Shah. Revisting of Elliptical Curve Cryptography for Securing Internet of Things (IOT), Advances in Science and Engineering Technology International Conferences (ASET). 2018.
26. B. Sabir, M. Youssfib, O. Bouattaneb, H. Allalia. Authentication and load balancing scheme based on JSON Token for Multi-Agent Systems, Second International Conference on Intelligent Computing in Data Sciences (ICDS), 2018.
27. J. R. Shaikh, M. Nenova, G. Iliev, Z. Valkova-Jarvis, Analysis of Standard Elliptic Curves for the Implementation of Elliptic Curve Cryptography in Resource-Constrained E-commerce Applications. IEEE International Conference on Microwaves, Antennas, Communications and Electronic Systems (COMCAS), 2017.
28. M.H. Shao, J. Zhou. Protecting mobile-agent data collection against blocking attacks. Computer Standards & Interfaces, 2006.
29. C. Zrari, H. Hachicha, K. Ghedira. Agent's security during communication in mobile agents' system, in Procedia Computer Science, vol. 60, no. 1, pp. 17–26, 2015.
30. Foundation for Intelligent Physical Agents." FIPA Agent Management Support for Mobility Specification". document number dc00087c, 2002.
31. Mathematical routines for the NIST prime elliptic curves, 2010.
32. The cryptographic hash function SHA-256, CRIPTOGRAFIA MAII – FIB.

AUTHORS PROFILE

Yousra BERGUIG member of Informatics, System Optimization Laboratory IbnTofail University Kénitra, Morocco. She works as computer system administrator. Her research interests include Wireless Sensor Networks, Network Security, Business Intelligent, Big Data, artificial intelligence, machine learning, cryptography and Smart-Cities, Phd student in Mobile-Agent and distributed systems security at Laboratory of Informatics systems and optimization at Ibn Tofail University, Kenitra, Morocco.

Dr. Jalal Laassiri member of Informatics, System Optimization Laboratory at Ibn Tofail University, Kénitra, Morocco. Laboratory Team Head he received his Ph.D. degree in computer sciences and engineering from University of Mohammed V, Rabat, Morocco. He is Member of the International Association of Engineers (IAENG), Dr. laassiri authored/co-authored many Journal articles, Conference Proceedings and Book Chapters. His research interests include Wireless Sensor Networks, Network Security, Business Intelligent, Big Data, Smart-Cities, Internet of Things, He joined the Faculty of Sciences of Kénitra, Department of Computer Science, Ibn Tofail University, Morocco, as a Professor in October 2010.

Sanae HANAOUI member of Informatics, System Optimization Laboratory Ibn Tofail University Kenitra, Morocco. Phd student Working on the security of distributed systems and AI (MAS Technology). Her research interests include Wireless Sensor Networks, Network Security, Cryptography, artificial intelligence, machine learning. Affiliated to Informatics systems and optimization Laboratory at Ibn Tofail University, Kenitra, Morocco.

Dr. Salah-ddine Krit is currently an Associate Professor at the Polydisciplinary Faculty of Ouarzazate, Ibn Zohr University Agadir morocco, he's Director of Engineering Science and Energies Laboratory and The Chief of Department of Mathematics, Informatics and Management. Dr. Krit authored/co-authored over 130 Journal articles, Conference Proceedings and Book Chapters. His research interests include Wireless Sensor Networks, Network Security, Smart-Homes, Smart-Cities, Internet of Things, Business Intelligent, Big Data, Digital Money, Microelectronics and renewable Energies.