Mobile Agent Security Based on Cryptographic Trace and SOS Agent Mechanisms

Sophia Alami-Kamouri, Nabil Moukafih, Ghizlane Orhanou, and Said Elhajji
Laboratory of Mathematics, Computer Sciences and Applications-Information Security
Faculty of Sciences, Mohammed V University in Rabat. Morocco
Email: Sophia.alami.kamouri, moukafih.nab, elhajji.said}@gmail.com; orhanou@fsr.ac.ma

***Abstract*—**Mobile agent technology quickly emerged in the field of distributed computing. Its use offers several advantages compared to the classic Client / Server model such as reducing network traffic, using disconnected computing, and offering more flexibility in application development, etc. Although mobile agent technology is considered as a powerful tool, it has some security problems that need to be overcome. Mobile agents, due to their mobility are vulnerable to attacks in a hostile environment. For this reason, it is necessary to put effective security techniques to protect them when they move from their home host to a new one. In this paper, we propose two security mechanisms to enhance mobile agent security. On one hand, we use a cryptographic trace to ensure mobile agent integrity and origin authentication, and on the other hand an SOS agent monitor's model is proposed to protect the mobile agent system against malicious hosts and DOS attacks. An implementation of the SOS agent approach will be presented.

***Index Terms*—**Mobile agent, security, SOS agent, cryptographic trace.

I. Introduction

A mobile agent is a software code that can move from one platform to another at different times, it can suspend execution on one platform and then resume on another platform to get closer to appropriate resources.

When the client gives a mission to achieve to an agent, the latter moves in the network to access services locally and complete his task. This works in 3 steps:

- The client activate the mobile agent and describe its mission,
- The mobile agent migrates to access the services, and execute its tasks,
- The mobile agent returns to its home with the mission result.

The flexibility of mobile agents, their autonomy, their mobility, their adaptability on the network make them powerful in solving complex problems. However, agent mobility and autonomy pose security issues in distributed environments. When an agent moves, it is crucial to ensure that it will be executed correctly and safely on the new visited system. Similarly, it is crucial to reassure the receiving system that there will be no risk of hosting a new agent.

In this paper, we focus on the security of mobile agents. The proposed security model is based on a bi-dimensional approach to provide a desirable mobile agents security issues in different levels. To meet the safety requirements, we proposed two security mechanisms:

- We have adopted the cryptographic trace to ensure mobile agent integrity and origin authentication during its migration from one platform to another. The visited platform could, once the mobile agent arrives, proceed with the verification of its cryptographic trace, to authenticate it and to retrieve the message.
- In addition, we proposed the SOS [1] agent model which aims to protect the agent against malicious hosts and DOS attacks and then ensure system availability. Its role is to watch over the mobile agent security by monitoring its movements through the visited platforms using a timer. If the agent performs its mission before the timer expires, it must send an acknowledgement to the SOS agent before moving to the next platform, confirming that is safe and its mission was successfully accomplished. Our SOS agent, unlike existing models, remains in the home platform, which is the most secure one. If the timer exceeded and no message was received from the mobile agent, the SOS agent identifies the actual visited platform as malicious and adds it to its blacklist, then sends a new agent with the accumulated data to terminate the mission.

By these two mechanisms, we are able to ensure mobile agent integrity, availability and origin authentication.

The paper is organized as follows: Section II studies the security issues facing Mobile Agents, the security threats and the security requirements. Section III exposes some existing approaches to protect mobile agents. Section IV describes our proposed model to protect our mobile agents. In section V, we implement our proposed SOS agent approach and test several scenarios to prove the validity and feasibility of our approach against malicious hosts and DOS attacks. Finally, the paper is concluded in Section VI with a discussion about the proposed approach.

Manuscript received August 23, 2019; revised February 10, 2020.

We choose the name SOS for this agent to show that it will be triggered as a distress signal and request assistance.

Corresponding author email: Sophia.alami.kamouri@gmail.com.

doi:10.12720/jcm.15.3.221-230

II. Mobile Agents Security Issues

A. Security Threats

The security problem is a hindrance to the expansion of this technology. A mobile agent can be targeted by several types of attacks since it moves from one system to another that may not be secure. That's why the security of mobile agent-based systems can be handled in four plans:

- Security between two agents,
- Security between the agent and its execution platform,
- Security between two machines,
- security between an agent and an external speaker.

Indeed, there are 4 categories of security threats [1]:

- **Agent against Agent Platform:** this category represents all the threats in which the agents exploit the security weaknesses of an Agent Platform or launch attacks against it. An incoming agent has two main lines of attack. The first is to gain unauthorized access to information residing at the agent platform; the second is to use its authorized access in an unexpected and disruptive fashion. Unauthorized access may occur simply through a lack of adequate access control mechanisms at the platform or masquerading as a platform-approved agent. Once access is gained, information can be disclosed or the platform-resident information, including instruction codes, may be altered.
- **Agent Platform against Agent:** represents all the threats in which the platforms compromise the security of the agents. A receiving agent platform can easily isolate and capture an agent and may attack it by extracting information, corrupting or modifying its code or state, denying requested services, or simply reinitializing or terminating it completely.
  It may be corrupted merely by the platform responding falsely to requests for information or service, or delaying the agent until its task is no longer relevant.
- **Agent against other Agents:** is the set of threats in which agents exploit the security weaknesses of other agents or launch attacks against other agents. An agent can target another agent using several general approaches. These include actions to falsify transactions, eavesdrop upon conversations, or interfere with an agent's activity. For example, an attacking agent can respond falsely to direct requests it receives from a target or even deny that a legitimate transaction has occurred.
- **Other Entities against Both:** represents all the set of threats in which external entities such as other agents and other platforms, threaten the security of both the internal agents and the agent platform. Even assuming the last ones have good behaviors, other entities both outside and inside the agent framework may attempt actions to disrupt, harm, or subvert the agent framework.
  The obvious methods involve attacking the inter-agent and inter-platform communications through masquerade, (e.g., through forgery or replay) or intercept.

B. Security Requirements

Generally, a secure mobile agent system must achieve the following security objectives:

- **Authentication:** is the process of verifying the identity of a user, device, or entity before allowing access to a system's resources to prevent it from faking or masking information. A mobile agent must authenticate to each visited agent system and therefore an agent system is able to decide whether it is a trusted agent. In the same time, the mobile agent must be able to authenticate the agent system.
- **Authorization or Access Control:** the process of granting or denying a request from a user or a program or entity after confirmation of authentication.
- **Confidentiality:** requires that during the exchanges in a system, the data must be protected against unauthorized disclosure and that only the entities to which they are entitled may access it. The mobile agent and the agent system must protect their private information against unauthorized access.
- **Anonymity:** the security policies of the agent platform and their audit requirements must be carefully weighed against Agents' expectations of confidentiality. In this case, the platform must maintain secret the identity of the agent with respect to other agents.
- **Availability:** includes the availability of data and services from a mobile agent so that legitimate users can access data and systems in a timely manner. This property ensures accessibility to resources and / or services as long as it is an authorized agent.
- **Integrity:** divided into two parts: data integrity and system integrity. Data integrity means that data must not be tampered with or destroyed in an unauthorized manner to maintain consistency. The integrity of the system means that a system should be free from unauthorized manipulation. In the context of mobile agents, the agent's route is a datum that requires protection against all forms of alteration.
  The agent platform must protect agents against unauthorized modification of their code, execution status and data and ensure that only authorized agents or processes make changes to the shared data.
- **Non-repudiation:** means that each user and entity must not deny the communication made later. To do this, important communication exchanges must be recorded to prevent refusals of part of a transaction. It relies on authentication to register the identities of the entities.
- **Assurance:** reasons for confidence that other security objectives (including integrity, availability, confidentiality and non-repudiation) are adequately achieved through a specific implementation. This includes a feature that works correctly, a sufficient protection against unintentional errors (by users or

software) and a sufficient resistance to intentional penetration or bypass [1]-[8].

- **Fairness:** no party can have any advantage over the other parties. Thus, mechanisms are needed to ensure a fair interaction of the agent platform in the electronic exchange.

We will focus, in this paper, on the confidentiality, integrity, non-repudiation and authentication.

III. Existing Approaches to Protect Mobile Agents

Mobile agents are subject to many threats as they move and execute on another environment that is controlled by another mobile agent platform different from the one that created it. This is why different security requirements must be generated in a system to achieve the security objectives.

The mobile agent's technology has several advantages, such as the ability of the mobile agent to move and migrate from one machine to another to get closer to remote resources. A mobile agent has the ability to clone itself so that it can run in parallel across multiple systems at the same time and also its ability to communicate with other agents to share their knowledge and expertise. Despite these advantages, if the security aspects are not taken into consideration, this technology can quickly become destructive.

The platform from which an agent originates or is created is known as the home platform and is usually the most reliable. Once the mobile agent migrates to another environment, this new environment is called the host environment and takes full control over the agent's code, data state, and execution state. This makes it difficult to protect mobile agents from malicious hosts and exposes them to multiple security threats.

In the following, we will describe some mechanisms and solutions proposed in the literature to protect mobile agents.

a) **Securing mobile agent based systems against malicious hosts:** in this article [2], the authors discuss in general terms the security of mobile agents against malicious hosts. This work provides a solution against DOS attacks launched by a malicious host that blocks a Visitor Mobile Agent and prevents it from continuing its itinerary.
This approach uses two mobile agents: a primary agent denoted PA and a shadow agent denoted SA. The mechanism uses an acknowledgement and timing mechanism to ensure that a mobile agent has visited a host in its itinerary and has gone safely to the next. A host is considered non-blocking if it allows the PA to continue its task and to leave safely to the next host. The SA suspects malicious action if it does not receive an acknowledgment within an appropriate time, after which it requests help from the primary host to identify the malicious host and takes corrective action.
When the local host identifies the malicious host, it sends a new instance of the security authority to a secure host to meet SA that carries a copy of the collected data. SA will reload the data collected in the blank PA. The newly loaded PA will continue its route by ignoring the malicious host.
In this work, the issue that arises is that the SA once it will migrate to another platform it can be an attack target.

b) **Using Secure-Image Mechanism to Protect Mobile Agent against malicious Hosts (SIM):** the Secure-Image Mechanism (SIM) proposed by [3] aims to protect mobile agents against malicious hosts, eavesdropping and alteration attacks.
The operation of SIM is as follows: a mobile agent in SIM migrates from one host to another following an itinerary to perform his task. The mobile agent is encrypted when it moves from one host to another to protect the mobile agent in communication channels. The mobile agent is decrypted when it arrives to the host. Therefore, some parts of the mobile agent may face security problems if the host is untrusted. From this point, the importance of SIM comes to protect all parts of the mobile agent in untrusted hosts.
SIM generates a secure image for the mobile agent before it arrives at hosts that are classified as unreliable hosts. If the next host in the agent's itinerary is untrusted, the agent visits the near Secure-Image Controller (SIC) which generates a secure image of the agent and sends it to the untrusted host. This protects the original agent from visiting malicious hosts.
The weak point of this solution is that the trusted and untrusted hosts should be known which is not always the case in distributed systems.

c) **On Mobile Code Security:** Sander *et al* [4] provide computing with features encrypted using non-interactive computer method, with CEF (Computing with Encrypted Functions) as a solution created for mobile code security requirements.
The purpose of this proposal is to encrypt functions so that their transformation can again be implemented as programs. The resulting program will consist of clear text instructions that a processor understand, but it will not be able to understand the function of the program.
Although some theoretical results related to the CEF have been produced near the computing with encrypted data, these results seem to be impractical as regards to their computer feasibility as well as their interactivity.

d) **Time Limited Blackbox Security: Protecting Mobile Agents from Malicious Hosts:** in this paper [5], Hohl introduces BlackBox security to protect Mobile code against malicious hosts by

generating executable code from a given agent specification. This generated code is executed by the host as a BlackBox, i.e. the host cannot modify or read it but can only execute it.

The use of Blackbox can be a major risk for the host who will run it, since the latter cannot have enough information about what it receives.

e) **Extending execution tracing for mobile code security:** in this article [6], the authors aim to protect the mobile agent's code against denial of service and state tampering attacks caused by malicious hosts. They proposed an approach based on the extension of the cryptographic trace mechanism.

This approach involves a trusted third party, verification server that undertakes the process of verifying traces on behalf of the agent owner. When an agent owner launches a mobile agent to a host platform, it creates a copy of the agent's code and state and forwards them onto a verification server designated by the host platform.

While the host executes the agent, it creates a trace of this execution simultaneously. Upon request of migration, the host then forwards this trace and the final agent state to the designated verification server, which ensures that the execution sequence is valid. Once a verification server receives an agent copy, it will be aware of the identity of the platform executing the actual agent.

It can thus implement a mechanism to ensure that a trace of the execution arrives from the required host within a reasonable time. This provides a way to safeguard against some forms of denial of service attacks.

The concern of the authors on this approach is to ensure that the traces, the code of the agent and the agent status propagated securely in their system, and that the traces are correctly associated with the corresponding agents. Among the disadvantages of this work, there is the high cost of cryptography. The researchers should try to find ways to reduce the cryptography cost of the protocol used without compromising security properties.

Table I below is a comparison between the existing solutions regarding the security mechanisms they used and the security objectives they achieved. We can see clearly that no solution is ensuring at the same time the confidentiality, the integrity, the non-repudiation and the availability.

TABLE I: COMPARATIVE TABLE BETWEEN THE EXISTING SOLUTIONS

Article :	Protection mechanisms	Confidentiality	Integrity	Non-repudiation	Availibility
[2] :	Acknowledgement, timing, primary & shadow agent	No	No	No	Yes
[3] :	Secure Image (SIM)	Yes	Yes	No	Yes
[4] :	Computation with encrypted functions	Yes	Yes	No	No
[5] :	BlackBox	Yes	Yes	No	Yes
[6] :	Cryptographic trace	Yes	Yes	No	No

After discussing related work about mobile agent's security their advantages and disadvantages, we will present in the next section our proposal for the safety of mobile agents and their environments to ensure system integrity, data confidentiality and non-repudiation, in addition to agent protection against malicious platforms and DOS attacks to ensure system availability.

IV. PROPOSAL OF A NEW MOBILE AGENT SYSTEM SECURITY MODEL

Our solution focuses on protecting agents from malicious agent attacks, malicious platforms and to ensure the integrity of the system. We therefore suggest a bi-dimensional approach to deal with security threats in our system. Our proposal describes two security mechanisms: the cryptographic trace and the SOS agent.

A. Proposed Cryptographic Trace

The security of the mobile agent and its code is paramount, as malicious agents can attempt to gain unauthorized access to the host, or malicious hosts can extract confidential information embedded in the agent and use it afterwards. Cryptographic trace or tracing execution consists of a sequence of statement identifiers of instructions and platform signature information. It is composed of a sequence of pairs (n, s), where n represents unique identifiers and s is the signature.

The signature of the platform is necessary only for the instructions that depend on interactions with the computing environment. For instructions that rely solely only on the values of internal variables, a signature is not required and, therefore, is omitted [9].

The cryptographic trace is one of the methods that ensure effective security and data transmission. We use the cryptographic trace so that local agent that created and sent the LightWeight (LW) agent can trace its movements and itineraries. Keeping this trace, we will know the path made by the latter and be sure that there is no agent identity usurpation.

In our proposal, we assume that each host has a private key denoted *Ks* and a public key denoted *Kp* that will be used during the encryption and decryption of messages and signatures. As show in Fig. 1, our LW agent migrates from the initial platform *H*0 to the platform *H*1, then to

*H*2... *Hn* to achieve its mission and returns to the original platform *H*0 which created it.

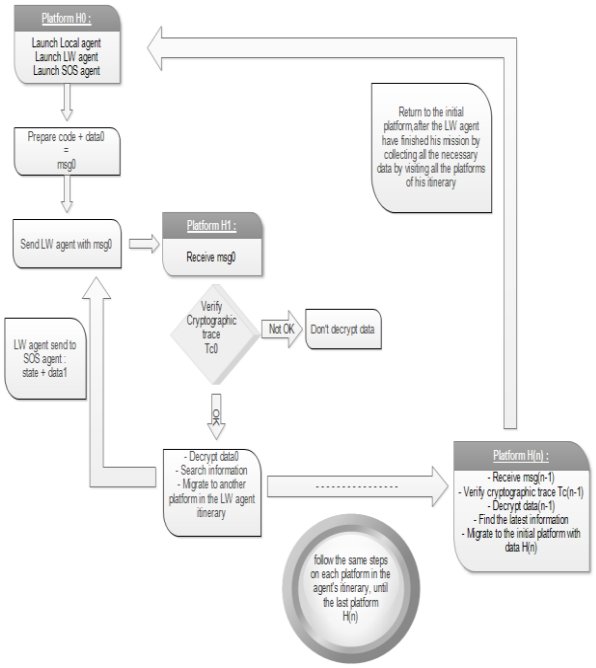

Fig. 1. Explanatory diagram of LW agent data transmission.

The message encapsulated by the mobile agent consists of a code to execute in addition to the identifiers, the cryptographic trace and the hash to ensure non repudiation and integrity.

Once the LW agent is launched and received his mission, it migrates **from platform** *H*0 **to platform** *H*1 with the following message:

$$msg0 = \text{id } H0, @\text{IP } H0, @\text{IP } H1, Hash0, Tc0, EncryptKp1(data0)$$

The sent message *msg*0 contains the following parameters:

- id *H*0 : the identifier of the transmitter platform.
- @IP *H*0 : the IP address of the transmitter platform.
- @IP *H*1 : the IP address of the receiver platform.

These first three entities specify that the message comes from *H*0 and is directed to *H*1.

- Hash0 = HashKs0(idH0, id LW agent) : the hash of the unique identifier of the LW agent and the identifier of the initial platform H0 with the H0 agent secret key.
  This Hash0 parameter is sent in each message, in order to be checked at the end of the agent's itinerary by the initial platform to show it is the LW agent that has been deployed in the beginning of the mission. Hash0 is also used to calculate the cryptographic trace.
- *Tc*0=*SignedKs*0(id*H*0, @IP *H*0,@IP *H*1, *Hash*0) : the cryptographic trace which is the sign of the id and the IP address of *H*0, *H*1's IP address and hash0 with the secret key of *H*0. This cryptographic trace shows that this message comes from platform *H*0 and destined to platform *H*1. Once H1 verifies this thanks to this cryptographic trace, the platform proceeds to the decryption of the data and the execution of the agent code.
- *EncryptKp*1(*data*0) : the data sent in the message is encrypted with the public key of the platform *H*1. This data will be decrypted by the private key of the platform that received the message, here *H*1.

Within the Platform *H*1 **:**

- Once the platform *H*1 receives the message *msg*0, it processes to the verification of cryptographic trace to know the source of message and whether it is destined to this platform or not.
- Cryptographic trace *Tc*0 as we saw above is the sign of the id and IP address of *H*0, *H*1's IP address and hash0 with the secret key of *H*0. So for the verification, the platform H1 decrypts this trace with the public key of *H*0 to make sure of the origin and destination of the message.
- When the cryptographic trace verification is conclusive, the platform processes the encrypted data sent in the *msg*0. The platform decrypts *EncryptKp*1(*data*0) with its private key and then the agent accomplishes its mission in this platform. Once the data processing is done, LW agent will send the result of the processed data (*data*1) to the SOS Agent with the message "go to the next", as we will see in the second part of our approach (next subsection).

LW agent migrates from platform *H*1 to *H*2 with message *msg*1 containing the following parameters: *msg*1 = id *H*1, @IP *H*1, @IP *H*2, *Hash*0, *Tc*1, *Tc*0, *Encrypt Kp*2(*data*1)

- The first three entities id *H*1 , @IP *H*1 , @IP *H*2 specify that the message comes from H1 and is directed to *H*2.
- *Hash*0 as explained above, it is sent in each message to be verified at the end by the original platform.
- *Encrypt Kp*2(*data*1) : data is encrypted with the public key of the platform *H*2. This data will be decrypted by the private key of the platform *H*2 that will receive the message.
- The cryptographic trace *Tc*1 = *SignedKs*1(id*H*1, @IP *H*1,@IP *H*2, *Hash*0, *Tc*0) is verified by the platform that receives the message.
- The cryptographic trace *Tc*0: is added in both Tc1 and msg1. It is added to the H1 signature to ensure the system integrity and it is added to msg1 to allow *H*2 to do the verification of the signature.

To generalize this process, we present the case of platform *Hi* and *Hi*+1 where $0 < i \leq n$.

Between Platform *Hi* **and** *Hi*+1**:**

The same steps outlined above are followed up to the platform *Hi*, where the message is the following : *msgi* = id *Hi*, @IP *Hi*, @IP *Hi*+1, *Hash*0, *Tci*, *Tci*-1, *EncryptKpi*+1(*data i*) and the cryptographic trace is : *Tci* = *Signed Ksi*(id *Hi*, @IP *Hi*, @IP *Hi*+1, *Hash*0, *Tci*-1)

When the LW agent completes his mission, it returns to the initial platform with the execution result.

Between Platform *Hn* and *H*0 :

Hn sends the following message *msg n* to *H*0 : *msg n* = id *Hn*, @IP *Hn*, @IP *H*0, *Hash*0, *Tcn*, *Tcn*-1, *EncryptKp*0(*data n*)

Within the Platform *H*0 :

- The Platform *H*0 checks the cryptographic trace *Tcn* to know the emitter of the message and the recipient, in the same way explained before:

$$Tcn = SignedKs0(\text{id}Hn, @\text{IP } Hn, @\text{IP } H0, Hash0, Tcn\text{-}1)$$

- Then, the second step is to check the *Hash*0. This parameter included in every message along the agent's itinerary that is verified once the initial platform receives the final message to ensure the integrity of the system.
- *Hash*0 = *HashKs*0(id*H*0, id LW agent) : the hash of the unique identifier of the LW agent and the identifier of the initial platform *H*0 with the secret key *H*0.
- When the platform receives the Hash0, which is a 32-bit value or a 64-bit value according to the hash function used, it calculates the expected one from the Identifier of the home platform and the identifier of the LW agent using its secret key. Then, it compares he received value with the calculated one.
  If both values are identical, the information and code sent haven't been changed or modified throughout the agent's itinerary.
- Then, *H*0 decrypts the received data *EncryptKp*0(*data n*).
  After presenting the proposed method for cryptographic trace to ensure the mobile agent integrity and origin authentication, in the next subsection, we will introduce a new mechanism to track the agent migration to avoid DOS attacks and to ensure system availability and the proper functioning of the agent.

B. SOS Agent Mechanism

In a typical multi-agent system, each platform uses two special agents: a Local Agent (LA) and a Lightweight Agent (LW). The Local Agent creates and assigns missions to the Lightweight Agent. The latter (LW) migrates from the home platform *H*0 to other platforms

H = {*H*1, *H*2 ... *HN}* as seen before, and looks for the desired information according to the assigned missions. Once the task is completed, the LW agent returns to the home platform. During its trip, one or more hosts in the specified itinerary could be malicious and would block the LW agent.

Even if we have proposed the cryptographic trace mechanism to ensure data integrity, to know LW agent itinerary and to be sure that there is no agent identity usurpation, we still need to ensure mobile agent availability by avoiding DOS attacks.

Indeed, our proposed model focuses on detecting denial of service attacks on LW agents when they migrate to perform an assigned task. Precisely, it uses a new agent called SOS agent which uses the same acknowledgement and time-out concept as used in the SA agent to monitor the movements of the PA agent in the paper [2]. The main difference is that, instead of lagging one or two steps in the itinerary behind the PA, our SOS agent will stay in the home platform, the most secure platform for a mobile agent, and monitor the movements of the LW agent. Using this approach, the SOS agent will not be a target of attack as it was the case with the shadow agent. When the LA assigns a task to the LW agent:

1) the LW agent will start its itinerary (*Hi*, i=0 ... N) while the SOS will stay in the home host *H*0.
2) When the LW completes its task in *Hi*, it sends an acknowledgement to the SOS agent along with the result of its execution in *Hi* *EncryptKp*0((*datai*)) and the accumulated data. The idea is that the LW agent sends an acknowledgement each time it finishes its task at a new host.
3) Next, the agent will move to the next one *Hi*+1.
4) If the SOS agent receives the acknowledgement, then we can assume that *Hi* is considered non-blocking, that the task has been executed safely and the agent can move to the next host *Hi*+1. In this case, the SOS agent will recalculates the timer based on the location of the next host in order to monitor the newly visited host and waits again for the new acknowledgement with the new accumulated data.
5) Contrarily, if the specified timer used by the SOS agent expires, and since we already know that *Hi*-1 is considered non-blocking because we have already received the previous acknowledgement (using the same logic), then we know for sure that the host *Hi* is malicious and necessary corrective measures must be taken.

With this approach, the SOS agent can exactly identify and blacklist the malicious host. That is, when the home host launches a new instance of the LW agent, the latter will receive the accumulated data from the SOS agent and migrate directly to the last visited non-blocking host *Hi*-1, but this time it will skip the malicious host according to its blacklist, the host *Hi*, and resume its trip in the remaining part of the itinerary starting with *Hi*+1.

It is important to note that since the SOS agent stays in the home host (the most secure host), the integrity of the accumulated data is protected from any modification or reverse engineering [11]. In addition, we have used a different interpretation of the acknowledgement and time out concept: The SOS agent uses a timer T to check whether the LW agent has executed its task safely in the current host before migrating to the next instead of checking whether the shadow agent will be blocked by one of the two destination hosts before performing the

task [2]. This difference gives the advantage of identifying and blacklisting exactly any detected malicious host.

Fig. 2 shows the operation of SOS agent:

1) Local agent, LW agent and SOS agent are launched, each one with its mission.
2) LW agent migrates to platform *Hi* to execute his task.
3) The SOS agent is waiting for the message from the LW agent using a timer T. After this step, two scenarios are possible:
4) Case1: If SOS agent receives the message "Go to the next" from LW agent and Encrypt(data). This means that its mission went well at current host. SOS agent recalculates the timer and waits again for its message.
5) Once LW agent sends "Go to the next" to the SOS agent, it migrates to the next platform to complete his task, and so on.
6) Case 2: If the timer has elapsed and the LW agent did not send a message, the SOS agent sends an alert to the local agent to deploy a new LW agent. If a late LW answer comes after the timer T, it will be ignored.

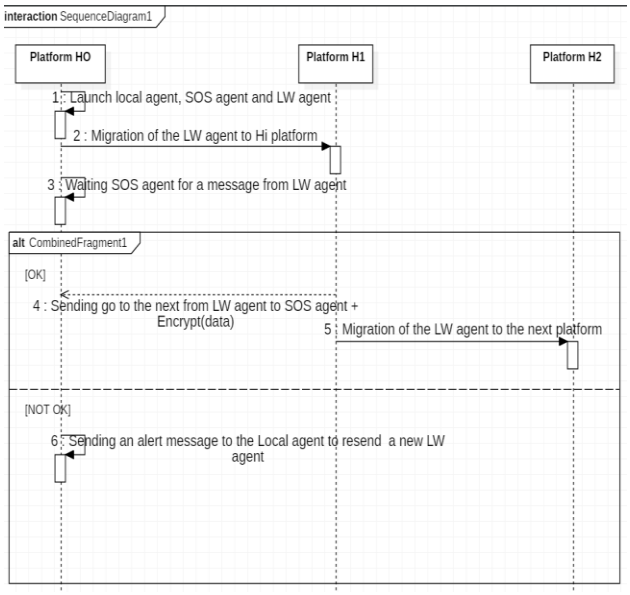

Fig. 2. Interaction between SOS agent and LW agent.

In the next section, we will describe the implementation and analyze in detail our SOS agent approach.

V. SOS AGENT APPROACH IMPLEMENTATION

The purpose of the SOS agent approach is to detect the denial of service attack by a malicious host that blocks a visiting mobile agent and prevents it from continuing its route/itinerary. To do this, we have implemented our proposed approach using Java Agent DEvelopment Framework or JADE 4.5.0 platform [12].

During the implementation, many scenarios were tested from the most trivial case where all hosts are considered non-blocking to end up with multiple malicious hosts. In all these scenarios, the SOS agent was able to detect all of the simulated malicious hosts, allowing the LW agent to skip all of them and only visit those who do not block. This gives confidence in the validity and shows the feasibility of the proposed approach.

A. DOS Attack Simulation

In order to simulate a DOS attack performed by a malicious host, we used ACL messages with specific responses and requests exchanged between the LW agent and a LA of each visited host.

- When the LW agent migrates to a host H*i* to execute an assigned task, it sends a "Hello" request to the H*i* Local Agent.
- If the host H*i* is malicious, then its Local Agent will respond with an ACL message with the content "Malicious", and then the LW agent will simply terminate. In a real case scenario, the LW agent will terminate due to a malicious action from the malicious host or platform.
- Otherwise (if the host H*i* is friendly), the H*i* Local Agent will send a "Friendly" ACL message, in this case the LW agent will execute normally and resume its itinerary thereafter.
- It is the role of the SOS agent to detect each and every malicious host in the environment.

B. Use Case Study

Following is a use case to explain the SOS approach operation. In this example, the LW agent moves in an itinerary of eight simulated hosts H1 to H8. As shown in Fig. 3, the hosts H2, H3, H5 and H7 are malicious hosts while the rest are non-blocking hosts.

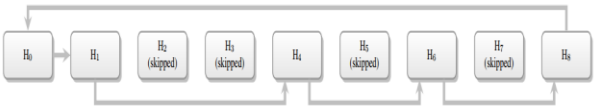

Fig. 3. Simulated environment.

In this example, the following scenario is simulated :

- In the home host H0, two agents SOS and Local Agent are initially created.
- The LA deploys a new mobile agent LW agent with a specific task to execute and some additional information (e.g. the itinerary). The SOS agent always stays at H0 and LW agent departs to visit H1.
- The SOS agent waits for an acknowledgement from the LW agent when it finishes executing in H1 and before departing to H2. The SOS agent uses a timer T that is calculated based on the location of the LW agent and the execution time of the assigned task. The time T should be long enough to receive an acknowledgement if there is no malicious action.
- Once the LW agent finishes its task, it will send an acknowledgement to the SOS agent, and moves to H2.
- If the SOS agent receives the acknowledgement before timer T expires, this means that host H1 is not

malicious host and LW agent is moving to H2. The SOS agent recalculates the timer T accordingly.

- The host H2 being malicious, it will block LW agent and terminate its execution. Therefore the LW agent cannot send the acknowledgement.
- When the timer T expires, SOS agent will decide that H2 has blocked the LW agent. In this case, it will send a request to the Local Agent stating that the LW agent is blocked in H2 and the last visited non-blocking host is H1.
- At this point, H0 will create a new LW agent instance with the new itinerary {H1, H3 ... H8} ignoring the malicious host H2.
- When the new instance of the LW agent arrives to the last visited non-blocking host, which is H1 according to the accumulated data, the LW agent will re-send the acknowledgement to SOS stating that H1 is non-blocking and it is moving to H3. In this case, the SOS agent will set its timer T to be ready for receiving the next acknowledgement when LW agent finishes its task in H3.
- The host H3 is malicious, so it will block the LW agent. The exact scenario will continue to detect and skip each malicious host until the LW agent eventually reaches H8 and returns to the home host H0.

Our approach has the main advantage of exactly identifying a potential malicious host using timers and acknowledgements. The SOS agent works according to the Algorithm 1 below:

Data: Blacklist, Whitelist, Itinerary
- Launch user interface of the SOS agent.
// All parameters are empty.
if *Button 'Deploy LW' is pressed* **then**
  // SOS agent will send in the request the itinerary {H_1, ... H_N} to use.
  - Send a request to Local Agent.
  // After deployment, The lightweight agent LW would autonomously migrate to the first host H_1.
  - Start timer and wait for 'Going to next: id of last container' as an acknowledgment.
  // Wait for confirmation before the timer expires.
  if *Acknowledgment is received* **then**
    // Add the last visited non-blocking host to the whitelist.
    - Update Whitelist.
    - Restart timer.
  end
  if *Timer expires* **then**
    - Update Blacklist.
    - Send a request to Local Agent
    // The request has the updated blacklist, updated whitelist, itinerary with last visited non-blocking host.
    - Restart timer.
  end
end

Algorithm 1. General Algorithm for the SOS agent.

The LA of H0 handles the deployment of the LW agents. It uses the steps shown in Algorithm 2.

Data: Blacklist, Whitelist, Itinerary
- wait for a request from SOS agent.
if *request is recieved* **then**
  - Update data.
  - Deploy Lightweight agent with new information.
  // In the first deployement (when no data is accumilated). the LW will start from H_1.
end

Algorithm 2. General Algorithm for the Local agent of H0.

We have customized our SOS agent with graphic user interface to monitor its behavior. This graphic interface provides the following information (see Fig. 4):

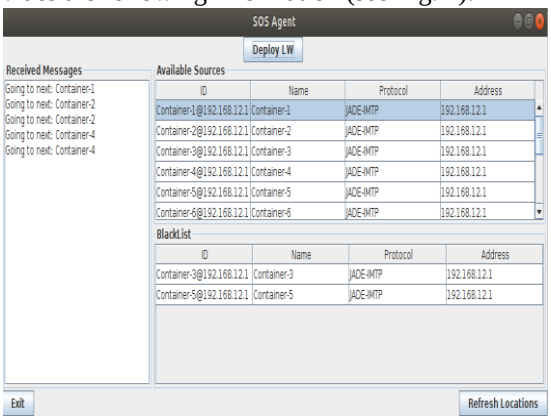

Fig. 4. Interface of the SOS agent.

- All sources / hosts available in the environment.
- A blacklist containing the list of malicious hosts detected so far.
- The contents of ACL messages sent by the LW agent (in this case it's: Going to next: the id of the last visited host.)
- A button to deploy the LW agent: when this button is pressed, the SOS agent will send a request to the LA of the home host H0 to deploy a new LW agent. This button is only pressed once.

```
Value of Dst is: null
Agent: LW0 is ready !
LW0 is now moving elsewhere.
Good hosts:[Container-1]
R.I.P Agent.
Malicious hosts: [Container-2]
Value of Dst is: Container-1
Agent: LW1 is ready !
LW1 is now moving elsewhere.
Good hosts:[Container-1]
R.I.P Agent.
Malicious hosts: [Container-2, Container-3]
Value of Dst is: Container-1
Agent: LW2 is ready !
LW2 is now moving elsewhere.
Good hosts:[Container-1]
Good hosts:[Container-1, Container-4]
R.I.P Agent.
Malicious hosts: [Container-2, Container-3, Container-5]
Value of Dst is: Container-4
Agent: LW3 is ready !
LW3 is now moving elsewhere.
Good hosts:[Container-1, Container-4]
Good hosts:[Container-1, Container-4, Container-6]
R.I.P Agent.
Malicious hosts: [Container-2, Container-3, Container-5, Container-7
Value of Dst is: Container-6
Agent: LW4 is ready !
LW4 is now moving elsewhere.
Good hosts:[Container-1, Container-4, Container-6]
Good hosts:[Container-1, Container-4, Container-6, Container-8]
Lightweight agent returned safely
```

Fig. 5. Console of the SOS agent.

In this use case, the simulated LW agent succeeded to visit all non-blocking hosts and skip all malicious ones. The LW agent started its journey from the home host H0 and successfully returned back to it as depicted in Fig. 5.

The Fig. 5 also shows that the SOS agent keeps track of the malicious hosts detected so far and also the list of the non-blocking hosts (labelled "good hosts"). Thus, whenever a LW agent is terminated by a malicious host, the SOS agent will deploy a new LW agent with the accumulated information as described before to help the new agent skip the malicious host.

VI. Conclusion

In this article, we focused on the security aspect of mobile agents when communicating and migrating to other hosts to get closer to remote resources. Our main goal is therefore to provide a desirable security to mobile agent-based systems. We have described the threats and security requirements faced by mobile agent technology. We have also presented some related work and highlighted some of their advantages and disadvantages. Although there are several security mechanisms and techniques to ensure the security of mobile agents, we found that still some important security issues are missing.

Next, we have introduced our proposed approach where we addressed important security requirements. To improve the security profile, we have taken into account attacks like DOS attacks. We have adapted the use of the cryptographic trace to guarantee the system integrity during the migration of the agent and to be sure that there has been no identity theft.

In our security approach, we also created an agent called SOS agent that is launched in the initial platform to monitor the LW agent when moving to another host. The SOS agent uses acknowledgements and a waiting period to avoid DOS attacks. This approach has been implemented and tested in a case study.

As we can see, the mechanisms that have been proposed ensure:

- The integrity of the system in addition to the authentication of the origin thanks to the use of the cryptographic trace which contains information about the emitter and the receiver of the message.
- The non-repudiation that is ensured by the signature.
- Data confidentiality using asymmetric encryption.
- The availability of the system and protection against DOS attacks thanks to the proposed SOS agent mechanism.

  Each time the agent migrates from one platform to another, the SOS agent launches a timer and waits for an acknowledgement from the LW agent that must send also the result of its mission in the actual platform.

  If the timer expires without receiving the LW agent acknowledgement, the SOS agent informs the local agent to start a new one and follow his itinerary.

In the following table, we give a comparison between the existing models and our proposed approach.

Table II: Comparative Table between the Existing Solutions and Our Proposition

Article :	Protection mechanisms	Confidentiality	Integrity	Non-repudiation	Availibility
[2] :	Acknowledgement, timing, primary & shadow agent	No	No	No	Yes
[3] :	Secure Image (SIM)	Yes	Yes	No	Yes
[4] :	Computation with encrypted functions	Yes	Yes	No	No
[5] :	BlackBox	Yes	Yes	No	Yes
[6] :	Cryptographic trace	Yes	Yes	No	No
Our approach	SOS agent, cryptographic trace	Yes	Yes	Yes	Yes

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

S. Alami-Kamouri proposed the new Mobile agent security model based on the cryptographic trace and the SOS agent mechanim.

N. Moukafih contributed to the improvement of the SOS agent mechanism and its implementation in JADE Plateform.

Pr. G. Orhanou and Pr. S. El Hajji supervised the research work, discussed the different security and technical aspects related to the proposed model, and give directions that helped to achieve the objectives.

All authors approved the final version and helped shape the research.

References

[1] W. A. Jansen, "Mobile agents and security," in *National of Standards and Technology Gaithersburg*, MD 20899, Special publication 800-19, USA, 1998.

[2] M. A. Madkour, F. E. Eassa, A. M. Ali, and N. U. Qayyum, "Securing mobile-agent-based systems against malicious hosts," *World Applied Sciences Journal*, vol. 29, no. 2, pp. 287-297, 2014.

[3] T. M. Ahmed, "Using secure-image mechanism to protect mobile agent against malicious hosts," *International Scholarly and Scientific Research and Innovation*, vol. 3, no. 11, pp. 364-369, 2009.

[4] M. Hefeeda and B. Bhargava, "On mobile code security," *Center of Education and Research in Information Assurance and Security*, 2001.

[5] F. Hohl, "Time limited blackbox security: Protecting mobile agents from malicious hosts," in *Mobile Agents and Security, Lecture Notes in Computer Science 1419*, Springer-Verlag, Berlin, 1998, pp. 92-113.

[6] H. K. Tan and L. Moreau, "Extending execution tracing for mobile code security," in *Proc. Second International Workshop on Security of Mobile Multi Agent Systems*, Italy, 2002.

[7] S. Srivastava and G. C. Nandi, "Fragmentation based encryption approach for self protected mobile agent,"

Journal of King Saud University – Computer and Information Sciences, vol. 26, pp. 131–142, 2014.
[8] P. Dadhich, K. Dutta, and M. C. Govil, "Security issues in mobile agents," *International Journal of Computer Applications*, vol. 11, no. 4, 2010.
[9] G. Vigna, "Cryptographic traces for mobile agents," in *Proceeding Mobile Agents and Security Springer-Verlag*, UK: London, 1998, pp. 137-153.
[10] S. Alami-Kamouri, G. Orhanou, and S. Elhajji, "Mobile agent service model for smart ambulance," in *Proc. 2nd EAI International Conference on ICT Infrastructures and Services for Smart Cities Brindisi*, Italy, April 2017.
[11] M. Popa, "Binary code disassembly for reverse engineering," *Journal of Mobile, Embedded and Distributed Systems*, vol. 4, no. 4, pp. 233-248, 2012.
[12] F. Bellifemine, G. Caire, and D. Greenwood, *Developing Multi-agent Systems with JADE*, Chichester, England Hoboken, NJ: John Wiley, 2007.
[13] N. Bouchemal and R. Maamri, "CAPMA: Clone agent to protect mobile agents in dynamic environments," in *Proc. International Conference on Advanced Aspects of Software Engineering*, 2016.
[14] C. Zraria, H. Hachichab, and Khaled Ghediraa, "Agent's security during communication in mobile agents system," in *Proc. 19th International Conference on Knowledge Based and Intelligent Information and Engineering Systems*, 2015.
[15] X. Vila, A. Schuster, and A. Riera, "Security for a multi-agent system based on JADE," *Computers and Security*, vol. 26, no. 3, pp. 91–400, 2007.
[16] L. M. Tsai and J. Tsai, "Formal modeling and analysis of a secure mobile-agent system," *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, vol. 38, no. 1, 2008.
[17] S. Hanaoui, J. laassiri, and Y. Bergui, "Security requirements and model for mobile agent authentication," *Smart Network Inspired Paradigm Approaches in IOT Applications*, pp. 179-189, July 2019.
[18] T. Bayer and C. Reich, "Security of mobile agents in distributed java agent development framework (JADE) platforms," in *Proc. Twelfth International Conference on Systems*, Venice, April 2017.

Sophia Alami-Kamouri, Phd student in Laboratory of Mathematics, Computing and Applications - Information Security, Faculty of Sciences, Mohammed V University in Rabat, Morocco. He received a Master degree in Cryptography and Security of Information from Faculty of science of Rabat, in 2014. Her research interests include the use of mobile agent paradigm in smart environment. She is interested also in information security.

Nabil Moukafih, Phd student in Laboratory of Mathematics, Computing and Applications - Information Security, Faculty of Sciences, Mohammed V University in Rabat, Morocco. He received a Master degree in Cryptography and Security of Information from Faculty of science of Rabat, in 2016. His main research interest is on event correlation and normalization using artificial intelligence approaches, SIEM, Mobile agent paradigm.

Ghizlane Orhanou, Associate Professor in Faculty of Sciences and member of the Laboratory of Mathematics, Computing and Applications – Information Security, Mohammed V University in Rabat, Morocco since 2013. She received PhD degree in Computer sciences from the Mohammed V University in Rabat in 2011 and the Habiltation to direct theses in 2016 from the same University. She received in 2001 a Telecommunication Engineer diploma from Telecommunication Engineering Institute (INPT-Morocco) and worked for about 3 years as a GPRS and Intelligent Network Engineer, and for 9 years as System and Network Security Engineer. Her main research interests include network and information systems security.

Said Elhajji, Professor. He graduated from Pierre and Marie Curie University (Paris VI, France) and received his PhD from Laval University in Quebec (Canada). He was associate professor at "Ecole Normale Supérieure" of Rabat then, and until 2018, he was professor at Faculty of Sciences, Mohammed V University in Rabat, Morocco. He was the director of the Laboratory of Mathematics, Computing and Applications-Information Security (LabMIA-SI) from 2005 to 2018, and also the responsible of the Master Cryptography and Information Security (CSI). His research interests include Modeling and Numerical Simulation, Numerical Analysis, Operating Systems and Networks Security, Information Security, Management of Information Security.