A Survey: Agent-based Software Technology Under the Eyes of Cyber Security, Security Controls, Attacks and Challenges

Bandar Alluhaybi[1], Mohamad Shady Alrahhal[2], Ahmed Alzhrani[3], Vijey Thayananthan[4]
King Abdulaziz University (KAU), Jeddah, Saudi Arabia

***Abstract*—Recently, agent-based software technology has received wide attention by the research community due to its valuable benefits, such as reducing the load on networks and providing an efficient solution for the transmission challenge problem. However, the major concern in building agent-based systems is related to the security of agents. In this paper, we explore the techniques used to build controls that guarantee both the protection of agents against malicious destination machines and the protection of destination machines against malicious agents. In addition, statistical-based analyses are employed to evaluate the level of maturity of the protection techniques to preserve the protection goals (the code and data, state, and itinerary of the agent), with and without the threat of attacks. Challenges regarding the security of agents are presented and highlighted by seven research questions related to satisfying cyber security requirements, protecting the visiting agent and the visited host machine from each other, providing robustness against advanced attacks that target protection goals, quantifying the security in agent-based systems, and providing features of self-protection and self-communication to the agent itself.**

***Keywords*—Agent; attack; cyber; security; requirement; maturity; protection goals**

I. Introduction

One of the most important software technologies that is used to manage and perform tasks over the Internet is agent-based software technology (ABST). A software agent is defined as an independent program that runs run on behalf of a network user [1, 2, 3]. ABST has been involved in many research fields, varying from network management tasks to information management ones [4, 5]. The power of the ABST is inspired by its valuable properties. The properties of this technology can be summarized as follows [3, 6]:

- Mobility, which is a unique property of this technology. It means that the agent can move from one machine to another machine, performing a specific mission there, and then it must come back to the original machine with the results. In other words, the agent is goal-driven.
- Adaptability, which means platform independent. In other words, this property enables the agent to be executed on different machines regardless of the operating system used.
- Transparency and accountability, which explains that the software agent runs on behalf its owner, and the owner of the agent can ask the agent about its current location and about what has been accomplished.
- Ruggedness, which refers to the capability of the agent to run on either low or high resources and to interpret different data formats.
- Self-start or proactive means that the time of starting a mission, the time of finishing a mission, and the time of delivering results are features that are based on the knowledge of the agent and have no relationship to the owner of the agent.

Thus, the agent is not restricted by the machine where it is written, but it has the ability of moving among machines via a network [7]. This action is called migration, as shown in Fig. 1.

In Fig. 1, different machines are connected via a network. The owner of the agent creates it at a machine called the home machine (HM). Then, the mobile agent can migrate to other machines called destination machines (DMs, where each DM has its own operating system (OS) as well as its own hardware (HW) specifications. A uniform agent manager, which is middleware (MW), is installed at the HM and at each DM. Concordia [8], Java Agent Development Framework (JADE) [9], and Agelets [10] are examples of agent managers. Fig. 2 shows a code example written by Concordia to illustrate the migration process of the mobile agent.

In the example above, the owner creates the agent, then creates an itinerary to move it to a DM called "dbserver", then back to an HM called "workstation". The agentsCodebase and relatedClasses specify the objects containing the methods and data necessary to complete the mission. More specifically, an itinerary is created, and when the agent ready to migrate, it prepares a list of its intended destinations. The itinerary of the agent is used by the Concordia server to determine the network destination of the agent. With each method included in the itinerary (i.e., "queryDatabase" and "reportResults"), the local Concordia server will move the agent and its objects to the machine specified in the nest itinerary entry. When the itinerary is exhausted, the trip of the agent is finished. Therefore, the itinerary caused the agent to move to "dbserver" and execute the "queryDatabase" method, then to move back to "workstation" and execute the "reportResults" method. The last line of the code illustrates an additional argument to the "launchAgent" method, which causes the codebase and the "QueryResults" class definitions to travel with the agent.

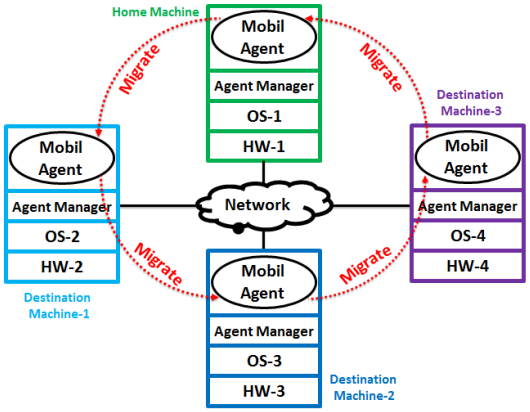


Fig. 1. Migration of Mobile Agent.

```
Public Class Test-Launch
{
    Public Static Void Main (String args [])
    {
        DBAccess-Agent Agent = new DBAccess-Agent ();
        Itinerary itinerary = new Itinerary ();
        itinerary.addDestination (new Destination ("dbserver" , "queryDatabase"));
        itinerary.addDestination (new Destination ("workstation" , "reportResults"));
        String agentsCodebase = "file:C:\MyAgent";
        String relatedClasses [] = {"QueryResults"};
        BootStarp.launchAent (agent, itinerary, agentsCodebase, relatedClasses);
    }
}
```

Fig. 2. Concordia Code for Mobile Agent Migration.

From the description above, four parts of the mobile agent are travelling during the migration. They are:

- The code of the mobile agent.
- The data, which are manipulated by the code.
- The itinerary information, which includes the HM and the DM.
- The state, which describes data controlled by the CPU and OS and includes the results of the executed mission.

Statement of the problem however, the mobile agents can be targets for attackers, where any one of the parts (or all of them) listed above can be the victim. This in turn can shoot the functionalities of any agent-based system in the heart. Specifically, passive attacks (such as eavesdropping and repudiation attacks) or active attacks (such as alternation and replay attacks) can be applied to the agents involved in the system. In passive attacks, the information carried by the mobile agent can be stolen to be misused later for malicious purposes [11]; meanwhile, in active attacks, the carried information is obtained and modified during the migration for the purpose of performing malicious actions [12]. The two previous kinds of attacks can be performed by an external attacker (i.e., located between the HM and the DM), but do not address the scenario in which the DM itself is the attacker. In this case, the danger may reach severe levels because the DM has full control of the execution of the hosted agent. On other hand, the mobile agent may itself be the attacker, with the ability of launching or performing poisonous pieces of codes against the DM. As a result, ensuring the security of the mobile agents as well as protecting the DMs against malicious agents is a pressing issue.

In this survey, we review the different techniques proposed previously to ensure the security in the software agent research field as well as the potential cyber-attacks. The contribution of this paper is as follows:

- We provide a statistical model called the maturity model to evaluate the protection mechanisms in agent-based systems. The maturity model relies on the protection goals, which represent the main parts of the mobile agent.
- We employ the maturity model for both evaluating the protection mechanisms under threats of different attacks and ranking the attacks according to their danger.
- We summarize the challenges of security of the agent's research field by seven research questions.

The rest of the paper is structured as follows. In Section II, we highlight the importance of agent-based software technology. Section III provides the cyber security requirements in agent-based systems. A classification of security techniques is presented in Section IV. Section V discusses achieving the cyber security requirements. In Section VI, the protection goals, attacks, and maturity model-based analyses are discussed. Section VII provides a strategy to evaluate an agent-based system with the security metrics that can be used. The challenges and the corresponding research questions are presented in Section VIII. Finally, we conclude the paper in Section IX.

II. Importance of ABST in Distributed Systems

Before the birth of ABST, many client-server-based technologies have been used for developing distributed systems, such as message passing (MP) [13], remote procedure call (RPC) [14], and Code on Demand (CoD) [15]. Under the interaction term between the client and the server, AGST overcomes the previous technologies, as shown in Fig. 3.

Fig. 3 shows that in MP, RPC, and CoD technologies, if a user wants to send (n) requests to the server, the network channel is occupied by n sizes of the requests. After processing the requests at the server side, the network channel will be occupied by n sizes of the responses. Formally, let $size_R$ denote the size of the request and $size_P$ denote the size of the response. BW denotes the band width of the network channel, and NT denotes the network traffic. Then,

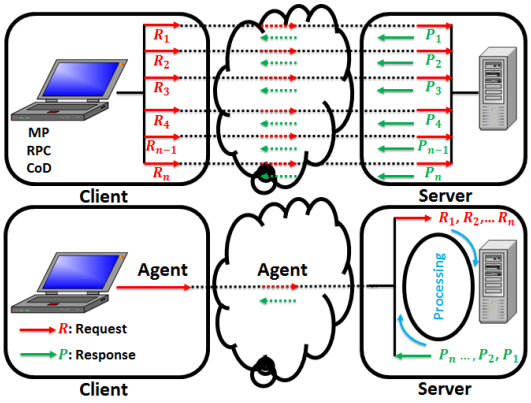


Fig. 3. ABST vs. other Technologies.

$$NT_{out-coming}\ \% = \frac{BW}{n \times size_R} \quad (1)$$

$$NT_{in-coming}\ \% = \frac{BW}{n \times size_P} \quad (2)$$

In the worst case, the interaction between the client and the server will be at a high level, which requires interleaving among the out-coming requests and the in-coming responses within the network channel. Thus,

$$NT\ \% = \frac{BW}{n \times (size_R + size_P)} \quad (3)$$

Compared to MP, RPC, and CoD technologies, the network channel is occupied by only the size of the migrated agent ($size_{agent}$), which is very small compared to ($n \times size_R$). This, in turn, contributes to reduce the network traffic efficiently. After processing the requests at the server side (i.e., executing the mission of the agent), the agent migrates back to the client, carrying the responses included in the state part. It is worth mentioning that there is no worst case in the ABST. However, during the migration back, the four parts (i.e., state, code, data, and itinerary information) are included within the agent. This will increase the network traffic slightly.

More formally, let ($size_{st}$), ($size_{co}$), ($size_{da}$), and ($size_{it}$) refer to the sizes of the state, code, data, and itinerary information respectively. Then, the size of the agent during migration ($size_{agent\_m}$) is defined as:

$$size_{agent\_m} = size_{co} + size_{da} + size_{it} \quad (4)$$

It is obvious that:

$$size_{agent\_m} < n \times size_R \quad (5)$$

During the agent's migration back and because of the results' size included in the state, the size of its state is:

$$size_{st} = n \times size_P \quad (6)$$

Thus, the size of the agent during migration back ($size_{agent\_mb}$) is updated to be:

$$size_{agent_{mb}} = (size_{co} + size_{da} + size_{it}) + (n \times size_P) \quad (7)$$

Because no interleaving exists when using agents, the size of the agent that migrates back is less than the sum of the sizes of the requests and responses of the worst case in other technologies. This is represented by the following formula:

$$size_{agent\_mb} < n \times (size_R + size_P) \quad (8)$$

Thus, the NT % based on ABST is less than the NT % based on other technologies.

Moreover, under the first aspect of the scalability quality attribute (i.e., increasing the number of users), the ABST also overcomes other technologies. Let M_{user} denote the number of users that are using a system, where each user sends n requests, each one of size $size_R$. Thus, the size of total number of sent requests ($size_{T\_R}$) is defined as:

$$size_{T\_R} = M_{user} \times n \times size_R \quad (9)$$

Compared to MP, RPC, and CoD technologies, each user creates an agent of size $size_{agent\_m}$ in the matched agent-based system. Consequently, the size of the total sent agents ($size_{T\_Ag}$) is:

$$size_{T\_Ag} = M_{user} \times size_{agent\_m} \quad (10)$$

When increasing the number of users (i.e., $M_{user} = 1000$, 2000, …, 10,000 users), it is obvious that:

$$size_{T\_Ag} \ll size_{T\_R} \quad (11)$$

Furthermore, under the second aspect of the scalability quality attribute (i.e., increasing the size of the manipulated data), the ABST is efficient. Let ($size_{mp\_da}$) refer to the size of the manipulated data at the server side. When increasing the size of the manipulated data, for example, such that ($k \times size_{mp\_da}$), where ($k = 2, 4, 6, \ldots, 10$), the performance will not dramatically deteriorate. This is quite true when dealing with Big Data (BD) sizes [16, 17]. That lack of deterioration is because the mobile agents migrate to the machines where the BD is located, processing it there, and then returning back with results of manipulation only. This provides an efficient solution to what is called the transmission challenge, which occurs because small sizes of codes (i.e., agents) migrate via the network channel to end tasks, rather than transmitting huge sizes of BD to the manipulating machines [18, 19, 20].

Since the time is tightly coupled with the transmission, ABST can overcome the network latency, especially when manipulating health data and multimedia [21, 22]. Moreover, under access latency and tuning time terms [23], the ABST outperforms other technologies. Access latency refers the time elapsed between the moment when a request is issued and the moment when it is satisfied. Let (T_{AL}, T_{comp}, $T_{sending}$, $T_{receiving}$) denote the access latency, computation time, sending time, and receiving time respectively. Then, access latency is defined as:

$$T_{AL} = T_{comp} + T_{sending} + T_{receiving} \quad (12)$$

Suppose that the T_{comp} is the same in both the ABST and other technologies. Because NT % based on ABST is less than NT % based on other technologies, both $T_{sending}$ and $T_{receiving}$ are short, which in turn leads to shorter access latency in the ABST. Tuning time (T_{TU}) is defined as the time a machine of a client stays active to receive the requested data. Since the T_{AL} is short in the ABST, the T_{TU} is also short compared to other technologies. This is quite true when the client uses his/her smart phone as a machine to send the requests and to receive the corresponding results, contributing to power consumption savings [24].

Table I highlights the characteristics of the ABST compared to other technologies.

Other benefits of agents, such as executing dynamically, asynchronously, and autonomously, are discussed in the work [25], where the authors provided seven good reasons for using the ABST.

TABLE. I. CHARACTERISTICS OF THE AGENT-BASED SOFTWARE TECHNOLOGY (ABST)

Term \ Technology		ABST	MP	RPC	CoD
Network traffic		Low	High	High	High
Scalability in big data	*Increasing NO. of users*	Efficient	Inefficient	Inefficient	Inefficient
	Increasing size of data	Efficient	Inefficient	Inefficient	Inefficient
Transmission challenge		Solved	Un-solved	Un-solved	Un-solved
Network latency		Low	High	High	High
Manipulation	*Access latency*	Short	Long	Long	Long
	Tuning Time	Short	Long	Long	Long

An additional feature is added to the ABST when comparing it to other technologies, such as component-based software technology (CBST) and web service-based software technology (WSBST). This feature is related to architecture building. In the CBST and WSBST, the architecture of the proposed distributed system is built during the design time. Meanwhile, the architecture is built during the run time in the ABST. Moreover, the ABST adopts the three types of architectures (i.e., sequential, parallel, and hybrid architectures). Actually, Fig. 1 above represents the sequential architecture, where the agent created at the HM visits a series of DMs in a sequential manner. Finally, the agent migrates back to the HM. Fig. 4 and 5 illustrate the parallel, and hybrid architectures, respectively.

In Fig. 4, the owner creates three different agents at the HM. Then, the agents are migrated in parallel to the corresponding DMs. In detail, the first agent migrates to DM-1 to perform its own task. The same scenario is followed by the second and third agent, where they migrate to DM-2 and DM-3 respectively. Finally, each agent migrates back to the HM with the results once its mission is finished.

Fig. 5 illustrates a hybrid agent-based architecture, where two different agents are created at the HM. The two agents start their itinerary in parallel, where the first agent visits DM-1 and DM-2, and then migrates back to the HM in a sequential manner. The second agent behaves the same, but its itinerary contains DM-3, DM-4, and DH-5.

Due to the benefits of the ABST explained above, it is involved in building a wide spectrum of distributed systems. Resource management in cloud computing [26], fault tolerance [27], distributed network performance management [28], security testing in web-based applications [29], and privacy protection in location-based services [30] are agent-based distributed systems, where agents play a significant role in performing the functionalities of these systems. However, again, the security of mobile agents at the interface remains a critical issue.

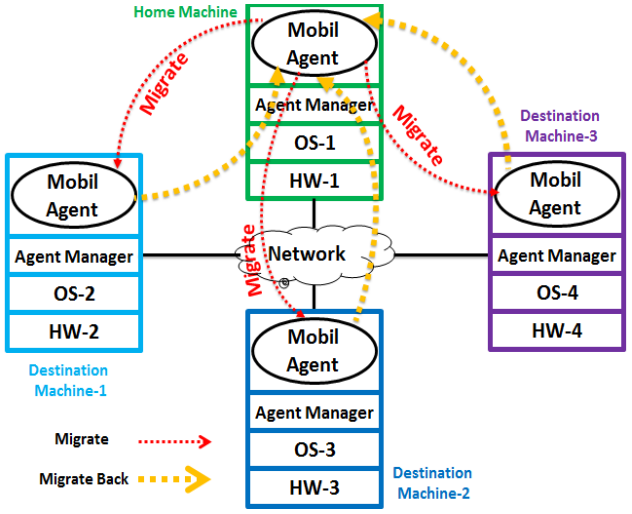


Fig. 4. Parallel Agent-based Architecture.

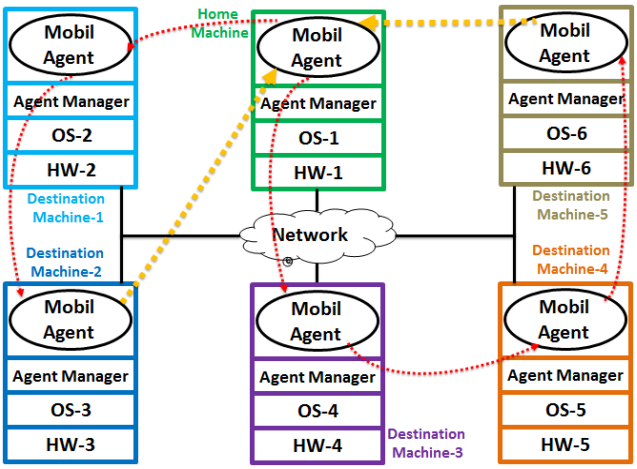


Fig. 5. Hybrid Agent-based Architecture.

III. CYBER SECURITY REQUIREMENTS IN THE LIFECYCLE OF MOBILE AGENT

This section explains the stages of the lifecycle of mobile agents and defines the cyber security requirements (CSR) needed to safely end the assigned mission.

A. Lifecycle of the Mobile Agent

There are three main stages involved in the lifecycle of the mobile agent, which are creation, migration, and termination, as shown in Fig. 6.

In the creation stage, the mobile agent is created (i.e., is written by the owner using a specific agent manager) with its itinerary as well as the mission that should be performed. This stage is conducted at the HM. In the migration stage, the mobile agent follows the path of the itinerary, visiting one or more DMs. After completing the itinerary, the agent is terminated. If the mobile agent safely returns to the HM, the termination is performed by the owner of the agent. Otherwise, it is killed or blocked by a visited DM (i.e., it is attacked). In other words, the danger to the mobile agent starts at the moment of leaving the HM, where it can be attacked during moving among DMs or by any of the visited DMs.

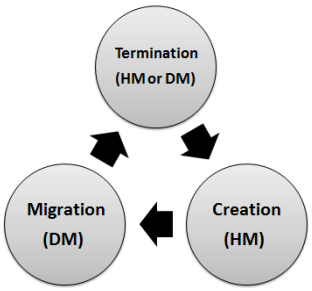


Fig. 6. Lifecycle of the Mobile Agent.

B. Cyber Security Requirements

If we want to represent cyber security, we can represent it as an umbrella under which two main aspects are located, which are: security and privacy. Ensuring security means establishing a secure communication between the sender and receiver to safely exchange messages, where cryptography is the core of the techniques used in this aspect. Meanwhile, privacy means protecting sensitive data against misuse by attackers. To highlight the importance of the privacy aspect, we can consider location-based services (LBS), for example, where the user sends a query asking for the nearest hospitals. In LBS-enabled applications, the user is forced to reveal sensitive data, such as the real location and the queried Point of Interest (PoI), which in turn reflects personal aspects in his/her realistic life, such as a religious or health state. Such sensitive data can be exploited and misused later by attackers for blackmail or mugging. Indeed, the authors of [31] and [32] provided surveys on the techniques used to protect the privacy of the user, where the dummy-based technique [32, 33] is considered a powerful approach for this purpose.

In distributed systems, the key security requirements are represented by the CIA triads (i.e., confidentiality, integrity, and availability); meanwhile, the key privacy requirements are represented by the TLI triads [34, 35] (i.e., tractability, linkability, and identifiability). Fig. 7 illustrates the security and privacy triads under the cyber security umbrella.

The CIA represents traditional security requirements for designing and implementing any distributed system. However, using the ABST demands additional security requirements, which are the Six A's (i.e., anonymity, accountability, authentication, authorization, accounting, and assurance) as well as non-repudiation and verification. Table II below describes the CSR needed in an agent-based distributed system.

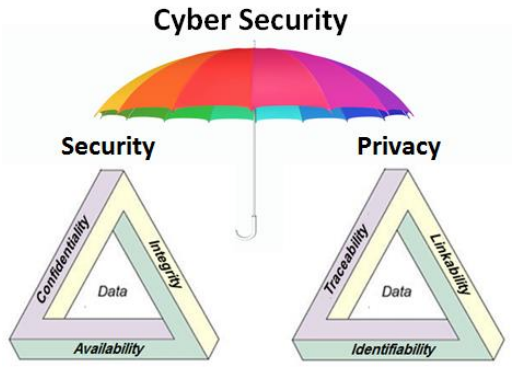


Fig. 7. Security Triads and Privacy Triads.

TABLE. II. CYBER SECURITY REQUIREMENTS

Aspect \ Term	Abbr./Name	Description
Security	***Traditional security requirements***	
	C (confidentiality)	The information carried by the mobile agent must be kept secret and only authorized parties can access it.
	I (Integrity)	Guarding the carried information against improper modification or destruction.
	A (Availability)	The assurance that the carried data are accessible when needed by authorized parities, including users and DMs.
	Six A's	
	An (Anonymity)	Achieving load balancing between keeping the actions of the agent private and auditing the agent when utilizing/logging the resources of the DMs.
	Ac (Accountability)	All actions that are performed on a DM should be traceable to the agent who committed them (i.e., logs should be kept, archived, and secured).
	Au (Authentication)	The positive identification of both the agent seeking access to a current DM and the carried information from a previous machine in an itinerary before execution of the mission on the current DM.
	Ar (Authorization)	The act of granting the agent actual access to information resources of the DM, where the level of access may change based on the agent's defined access level.
	At (Accounting)	The logging of access and usage of the DM's resources. In other words, keeping track the agent who accesses what resource, when, and for how long.
	As (Assurance)	The controls used to develop confidence that security measures are working as intended. Auditing, monitoring, testing, and reporting are the foundations of assurance.
	Additional security requirements	
	Non-R (Repudiation)	The agent platform that sends the information to an agent owner or other DM cannot deny that he is the owner of the specific information and agent.
	Ve (Verification)	Only the authenticated mobile agent is permitted access into the DM, and the code of the migrated agent from the HM is verified before execution.
Privacy	T (Tractability)	The ability to verify the history, location, or application of an agent by means of documented recorded identification.
	L (Linkability)	The attacker can sufficiently distinguish whether two or more agents are related or not within the system.
	I (Identifiability)	The attacker can sufficiently identify the entities within the system.

Both security and privacy must be considered as top quality attributes in designing agent-based distributed systems. Consequently, the CSR mentioned above should be satisfied over all stages of the agent's lifecycle, to ensure that the system is perfect under the cyber security term.

IV. Classification of Security Techniques in Mobile Agents

There are two main classes of approaches proposed in the research field on the security of mobile agents, which are the approaches that secure the agent platform (i.e., DM) against malicious mobile agents and the approaches that secure the mobile agents against malicious platforms. Each class has its own techniques, as shown in Fig. 8.

There are two main classes of approaches proposed in the research field on the security of mobile agents, which are the approaches that secure the agent platform (i.e., DM) against malicious mobile agents and the approaches that secure the mobile agents against malicious platforms. Each class has its own techniques, as shown in Fig. 8.

A. First Class: Protecting the Agent Platform

In this class, the attacker is the malicious mobile agent that visits a DM, and the victim is the DM platform. Many techniques have been proposed to protect the DM platform as described below.

1) Sandboxing: This is a software technique that depends on the principle of isolation of the execution of the suspected code in a virtual space under tight restrictions. Relying on the sandboxing technique, the authors in [36] proposed a mechanism that enforces the mobile agent to follow a fixed security policy for execution its code. This mechanism succeeds in preventing the mobile agent from (1) interacting with the local file system; (2) accessing the system properties; and (3) opening a network connection. Under this technique, an enhanced approach was proposed by Noordende et al. in [37]. The authors focused on the restrictions that deal with memory to prevent the unauthorized access by the poisonous code.

However, the major drawback of the sandboxing technique is that it consumes a long execution time (due to the strict restrictions) even if the mobile agent's code is legal.

2) Code signing: This technique targets ensuring the integrity of the code that is executed on the DM platform. It tunes with both the one-way hash functions and the digital signature concepts to ensure that no modification is done on the code. Therefore, this technique assumes that the creator of the code is trusted. The authors of the work [38] provide shining proof about the resistance of this technique, where it is used in ActiveX controls and Java applets. An enhanced verification-based approach is introduced by Malik et al. [39]. Their key idea depends on using white and black lists of entities, where a security manager checks the incoming code. If it is coming from a trusted entity (i.e., included within the white list), the code is then granted full permissions to be executed. Otherwise, it will be frozen.

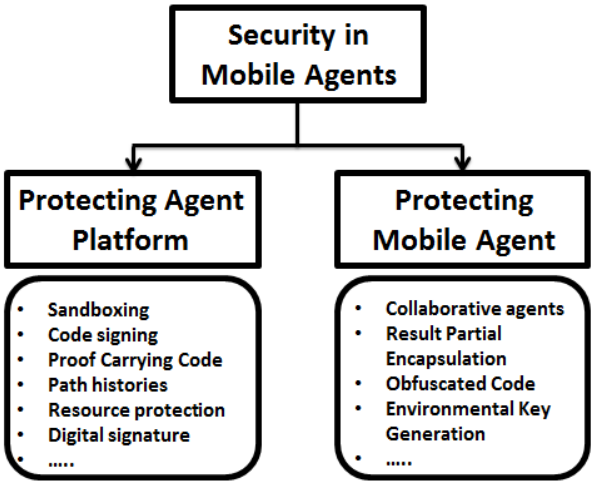


Fig. 8. Classification of Security Approaches in Mobile Agents.

However, the main drawback of this technique is that it requires the continuous update of both white and black lists, which is a large obstacle in light of the changing dynamic nature of entities as time progresses. In addition, it is computationally costly due to using hash functions in addition to encryption and decryption [40].

3) Proof Carrying Code (PCC): In this technique, the creator of the code marks the code (i.e., generates a proof attached to the original code), so that any modification that occurred will be detected and the code is not allowed to be executed. Compared to the code signing technique, the PCC is better regarding time and computation costs. The reason behind this is that the PCC does not require cryptography for the digital signature. In the work [41], the authors proposed a foundational proof-carrying code, in which the code is verified with the smallest possible set of axioms, using the simplest possible verifier and the smallest possible runtime system. An enhanced PCC-based technique is presented in [42], where the major concern is allowing dynamic access to the platform of DM, with a tolerance of the strict proof representation.

However, a sharp criticism was directed at this technique in [43], where the proof generation is the main problem with PCC, as well as the automation of this process.

4) Path histories: This technique embraces the principle of ascertaining the level of trust of the visited DMs' platforms during the life cycle of the agent. Therefore, the mobile agent is forced to maintain an authenticable trajectory of the previously visited DMs. Relying on this technique, the authors of the works [44] and [45] proposed two approaches that grant the mobile agent suitable privileges that match the corresponding trust levels.

However, the problem of this technique is explained in [46]. The cost of checking the trust level increases when the number of visited DM involved in the path history is increased. Moreover, it is complex to predict the trust level of the DMs visited in the future that are included in the path history in advance.

5) Resource protection: This technique employs the fundamentals of authentication to allow only legal agents to access the resources of the DM. Thus, the platform of the agent at the DM is protected. An authentication-based proxy is proposed in [12], where the mobile agent is not allowed to access resources unless it reveals its identity using the public and private keys. Another approach is presented with this technique in [47]. The authors' idea was inspired from the real-world scenario, in which the mobile agent can prove its identity by providing a passport and visa, which carries information that describes the credibility of the agent.

However, the limitation of this technique is related to the proxy overhead computation. In addition, the identity (i.e., the passport) may be stolen or impersonated.

6) Digital signature: It is a common technique used in secure communication networks that satisfies confidentiality and integrity. It is similar to the code signing technique, but the difference is applying a digital signature on the mobile agent itself instead of the carried code. A digital signature-based approach supported by a checkpoint mechanism is provided in [48]. The objective of the checkpoint mechanism is to guarantee the validity of the mobile agent using fragmentation and defragmentation methods. Based on both the digital signature and verifying method, another approach is proposed in [49]. In this work, the authors mix the code signing technique with the digital signature technique. The code of the agent is signed by the creator, and the code is executed at the DM after being verified by the owner of the agent.

However, supporting the digital signature-based technique by fragmentation and verification leads to a trade-off between the strength of the proposed approach and the computation cost.

7) Policy-based model: In this technique, predefined diagnosis methods are applied to the mobile agent once reaching the DM. Based on the results of the diagnosis, the agent is allowed or not allowed to execute. A malicious content scanning-based approach is presented in [50]. The scanner provides an alarm to the DM if any suspected content exists. An immune system is proposed in [51]. Actually, the work [51] is considered a development of the work [50], where performance was the axis of the enhancement. The key idea is to employ the pipelining concept in scanning, predicting, and extracting the malicious piece of code.

However, although the performance is enhanced, the process of scanning and discovering the malicious content is still costly due to the different u of the mobile agents' executions.

8) State appraisal. This technique tunes with the state carried by the mobile agent in a pure programming way. In depth, a maximum set of safe permissions that the agent could request from the DM is encapsulated within a state appraisal function, depending on the agent's current state. Based on this technique, a state appraisal function is proposed in [52] to ensure the security of the DM. The agent calls the state appraisal function to retrieve the permissions of the current visited DM and does not violate them. Then, when the mobile agent leaves the current DM moving to the next one, the state appraisal function is called again. Thus, the previous state, which represents the input of the mission that should be performed at the next DM, is ensured. In this way, the next DM guarantees that the state was not modified, and consequently, the arriving agent is not malicious. Similar to [52], the authors of [53] rely on the state appraisal function, but the difference is supporting the function by an authentication mechanism between the sender of the mobile agent from the current DM and the receiver of the mobile agent at the next DM.

However, the major issue in this technique is the difficulty of formulating and adopting the mobile agent with the security permissions of each visited DM.

9) Machine learning: This technique employs data mining concepts to protect the visited DM, depending on a classification data mining task. Recently, a supervised machine learning classifier was proposed in [54] by Pallavi et al. The authors used a data set that contains 80 mobile agents (half of them are malicious, and the remaining are non-malicious). Then, the features of all agents are extracted to determine the behaviors of the agents. Finally, using the extracted features, a decision tree-based algorithm is applied to the data set to make the execution decision related to a mobile agent. Depending on the data mining classification task, another approach is introduced in [55]. The same strategy used in [54] is used in [55], but the difference is that the authors used the K-nearest neighbor algorithm to build the classifier instead of the decision tree-based algorithm.

However, the main obstacle encountered with this technique is the excessive expense of building a good knowledge data base with a large number of agents. In addition, using different classifiers leads to different results.

Second Class: Protecting the Mobile Agent

In this class, the attacker is the visited DM and the victim is the mobile agent. Many techniques have been proposed to protect the mobile agent against the DM, as described below.

10)Collaborative agents: The principle of this technique relies on sharing secret information about the sensitive tasks between two cooperating agents, so that the DM cannot steal and tamper with the trajectory of the itinerary. Depending on this technique, a secure communication protocol between the agents is proposed in [56]. This protocol establishes an authenticated communication channel between the cooperating agents to share the content of the itinerary of the first agent with the second agent. The content of the itinerary adjusts the triads of visited DMs (i.e., the previous/last visited DM, the current DM, and the future DM). The second cooperating agent takes the responsibility for manipulating any inconsistency that may occur, such as the current DM sending the agent to the wrong future DM or generating an alarm about receiving the

agent from a wrong source. Madkour et al. proposed another cooperative approach to protect mobile agents against malicious DMs in [57]. The key idea is to create an assistant agent, called the shadow that follows the original one. If the original agent is attacked by the DM, it informs the shadow, kills itself, and the shadow in turn sends an acknowledgement to inform the owner of the original agent about the attack. The shadow then becomes the original agent, and the owner of the agent creates a new agent to be a new shadow.

However, the gap of this technique is the cost of configuration and establishing the authenticated communication channel for each migration.

11)Result Partial Encapsulation (RPE): This technique is designed to detect any changes that might occur regarding the results of an executed mission at a DM by a mobile agent. To end this, the results are encapsulated so that a verification step is performed later at the HM to provide proof that no change was performed by an attack. This technique is applied on the agent's code to provide confidentiality using encryption based on the secret key [58]. The key idea is to have a list of secret keys stored within the mobile agent, used for encryption, such that each key is related to a specific DM. In the current DM, the agent uses the corresponding secret key to generate message authentication code (MAC). Then, encapsulating the MAC with a message that represents the results of the mission execution generates partial result authentication code (PRAC). Based on the RPF technique, the authors of [59] proposed an approach to ensure both confidentiality and integrity of the results using a digital signature. This approach is called sliding encryption, which aims at decreasing both the time processing and the required storage by encrypting a small amount of data. The sliding encryption approach is developed so that it can be adopted in certain applications where storage space is valuable, such as smartcards.

However, the main drawback of this technique is ensuring future integrity, where the next DM can obtain the secret key of the previous DM to modify its generated results.

12)Obfuscated Code: In this technique, the mobile agent travels through series of DMs that have different trust levels. To ensure that no DM is able to extract sensitive data hidden in the code (such as the secret key or credit card number), the behavior of the mobile code is protected. The core protection performs some obfuscating transformations on the code before actual execution, so that the code cannot be understood by the malicious DM. Based on the obfuscation code technique, Hohl et al. [60] proposed the black box security approach to preserve the behavior of the code. They obfuscated the data structure used within the code without modifying the code itself. Another approach is provided in the context of this technique in the work [61]. The difference here is the way of modifying the code, where the control flow in the code is modified without affecting the computing part of the code.

However, the main challenge of this technique is adopting it to suit different applications, where behaviors can extensively change from one application to another.

13)Environmental Key Generation: This technique relies on the principle that states "the execution is not allowed unless some environmental conditions are satisfied at the DM". In the work [62], the authors defined the environmental conditions as matching a specific search string. When this condition is true, an activation key is performed to allow the execution. The activation key function is hidden within a file system. Similar to [62], the authors of the work [63] used the same condition, but the difference is that the activation key is included within the content of an email.

However, the limitation associated with this technique is that the DM may act maliciously after the condition is satisfied and the activation key is performed. Moreover, the key activation may be a virus file to be executed at the DM. Therefore, the DM tries to not allow execution even if the condition is satisfied.

14)Execution tracing: This technique targets discovering malicious modifications that may be performed by DM on the mobile agent code, state, and execution flow. The scenario followed by this technique consists of three steps, which are (1) a DM that receives the agent and agrees to execute it and produce an associated trace during the agent's execution; (2) a message is attached by DM to the mobile agent, containing information about the unique identifier of the message, the identity of the sender, the timestamp, the fingerprint of the trace, and the final state carried by the agent; and (3) the HM (i.e., the owner of the agent) asks the DM to provide the previous message (the trace) to validate it by comparing it with a fingerprint generated by the agent. In [64], a detailed protocol for message exchanging is provided to adjust the previous three steps in a mathematical manner. An enhancement is achieved on the previous protocol by Tan et al. in [65]. The key enhancement is assigning the mission of the trace validation to a trusted third party (TTP) instead of the owner of the agent. The TTP here is called validation or verification server.

However, the execution tracing technique suffers from the potential malicious collaboration between the validation server and a DM.

15)Watermarking: Originally, the watermarking term refers to the process of embedding a watermark within an information entity, such as image, audio, video, or text files for copyright protection purposes. The authors of [66] exploit the watermarking technique to detect an attack that aims at modifying the results of the mobile agent's mission execution. Consequently, the results are watermarked, and if a DM attacked them, the embedded watermark is damaged or destroyed. To detect the occurrence of the attack, the watermark is extracted at the HM and compared with the original one. The work [66] is developed by the same authors in [67] to be adopted with various kinds of watermarks. Therefore, during execution, the agent can employ any kind of

available information as a watermark, such as dummy data, input data, intermediate variable values, or data originating from communications.

However, the watermarking technique has a critical gap, which is that the embedded watermark can be destroyed by a compression attack. Compression attacks can be performed by any external attacker (i.e., not by the DM). Thus, the DM is considered malicious while it is not.

16)Co-signing: This technique relies on hiring an external trusted party to co-sign the migration of the agent. In [68], the preceding DM is considered the external party, which acts as an observer by taking the responsibility of co-signing the mobile agent. Actually, the work [68] is proposed to give mobile agents resistance against multiple colluded DMs that target poisoning the results of execution. Another approach is presented in [69] based on the co-signing technique. The key idea is that after producing the results, the DMs encapsulate them with the information of the mission carried by the mobile agent. Then, the entire encapsulated package is encrypted and sent to the next DM at the same time. When the mobile agent reaches the next DM, a comparison is performed between the generated results and the mission information to discover any attack that may have occurred.

However, time consumption, network overhead, and robustness against Denial of Service (DoS) attacks are considered the main challenges in this technique, especially when the mobile agent carries a time-sensitive task.

17)Separation of privileges: The essence of this technique is managing the agent-based system by separating the tasks and assigning them to some major agents. The goal of this technique is to minimize the capabilities of the malicious DMs to attack the visiting agents. In [70], three agents control the system, which are controller agent (CA), worker agent (WA), and itinerary register agent (IRA). The CA is responsible for storing and manipulating the core data. The WA is responsible for storing and manipulating functions that have less importance than the previous one. The IRA is responsible for storing the addresses of the visited DMs, and the time at which the execution is performed on each DM. The authors of [71] followed the same strategy as that presented in [70]. The difference is that the privileges are supported by roles.

However, the process of extracting the privileges, separating them, and supporting them with accurate roles that control different cases that may be involved leads to increasing the complexity of the agent-based system.

18)Fragmentation-based encryption: This technique aims at enhancing the performance, where only the sensitive data that may be exploited by a DM are first extracted. Then, these sensitive data are encrypted. Finally, the encrypted sensitive data are randomized so that only the agent knows the process of backing the correct order. In [72], the bytes of the agent's code are scanned, and then the sensitive parts are encrypted and inserted within predefined arrays. When execution at the DM occurs, the agent uses the same randomization key (i.e., the seed) to retrieve the correct ordering of all code bytes. Similar to [72], the protocol proposed in [73] depends on a fragmentation technique. The difference is that the extraction, encryption, and randomization stages are performed by a TTP.

However, despite the performance enhancement achieved through encrypting only the sensitive data of the agent's code, the process of generating the seed of the randomization algorithm, applying the algorithm, and reordering the randomized code may lead (in some cases) to exceeding the time needed for encrypting all of the agent's code.

V. Achieving Cyber Security Requirements

Table III (a, b) below compares the approaches discussed in the previous section in terms of satisfying the CSR.

Drawn conclusions from Table III (a, b), the following observations can be made:

1) Among the six A's, anonymity and accountability security requirements are not achieved in all approaches related to protecting the agent platform. Actually, there is a clear and strong trade-off between these two security requirements, as it is obvious from their concepts. Anonymity and accountability security requirements are critical for the second class (i.e., protecting the mobile agent).

2) Linkability and identifiability privacy requirements are not achieved in all approaches related to protecting the agent platform. Therefore, the attacker (malicious agent) has the ability of revealing some entities of the system within the DMs. For example, in [50] and [51], the policy file used for protecting the agent's platform is known by the visiting agent. The policy file may reflect the sensitivity level of the system installed on the DM where the mobile agent is executed.

3) Among CIA, the availability security requirement is not achieved in all approaches related to protecting the mobile agent class, which is normal. Actually, the availability security requirement is critical for protecting the agent platform class, where not achieving it means that the DMs suffer from the DoS attack.

4) All of the six A's are critical and should be achieved in any approach related to protecting the mobile agent class.

5) The non-repudiation and verification additional security requirements are not achieved in all approaches related to protecting the mobile agent class. Actually, these additional security requirements are critical for protecting the agent platform class, so that the code of the agent is verified before execution and the HM cannot deny creating the mobile agent and sending it to the DMs.

6) All privacy requirements (TLI) were completely ignored and were not addressed in all approaches related to protecting the mobile agent class.

7) The authorization, accounting, and assurance security requirements are mandatory necessities and should be satisfied by both classes.

TABLE. III. (A) SYMBOLS

Symbol	**C**	**I**	**A**	**An**	**Ac**	**Au**	**Ar**
Based on	Confidentiality	Integrity	Availability	Anonymity	Accountability	Authentication	Authorization
Symbol	**At**	**As**	**Non-R**	**Ve**	**T**	**L**	**I**
Based on	Accountability	Assurance	Non-Repudiation	Verification	Tractability	Linkability	Identifiability

(B) SATISFYING THE CYBER SECURITY REQUIREMENTS

Class	**Tech** \ **CSR**	**Security aspect**											**Privacy aspect**		
		CIA			***Six A's***						***Add. SR***		***TLI***		
		C	I	A	An	Ac	Au	Ar	At	As	Non-R	Ve	T	L	I
Protecting agent platform	***Sandboxing***														
	[36]	√	√	√	×	×	√	×	√	×	×	×	√	×	×
	[37]	√	√	√	×	×	×	×	×	×	×	×	√	×	×
	Code signing														
	[38]	√	×	×	×	×	√	×	×	×	×	×	√	×	×
	[39]	√	×	×	×	×	√	×	×	×	×	√	√	×	×
	PCC														
	[41]	√	×	×	×	×	×	×	×	×	×	√	√	×	×
	[42]	√	√	×	×	×	×	×	×	×	×	√	√	×	×
	Bath history														
	[44]	×	√	×	×	×	√	√	×	√	√	×	×	×	×
	[45]	×	√	×	×	×	√	√	×	√	√	×	×	×	×
	Resource protection														
	[12]	√	√	√	×	×	√	×	√	√	×	√	×	×	×
	[47]	√	√	√	×	×	√	×	√	√	×	√	×	×	×
	Digital signature														
	[48]	√	√	×	×	×	√	×	×	×	×	×	√	×	×
	[49]	√	√	×	×	×	√	×	×	×	×	×	√	×	×
	Policy-based model														
	[50]	√	√	√	×	×	×	×	√	×	√	√	×	×	×
	[51]	√	√	√	×	×	×	×	√	×	√	√	×	×	×
	State appraisal														
	[52]	√	×	×	×	×	×	√	×	×	×	×	√	×	×
	[53]	√	×	×	×	×	√	√	×	×	×	×	√	×	×
	Machine learning														
	[54]	√	√	√	×	×	×	×	×	×	×	×	×	×	×
	[55]	√	√	√	×	×	×	×	×	×	×	×	×	×	×
Protecting mobile agent	***Collaborative agents***														
	[56]	×	√	×	×	√	√	×	×	×	×	×	×	×	×
	[57]	×	√	×	×	√	√	×	×	×	×	×	×	×	×
	RPE														
	[58]	√	×	×	×	√	×	×	×	×	×	×	×	×	×
	[59]	√	√	×	×	√	√	×	×	×	×	×	×	×	×
	Obfuscated code														
	[60]	√	×	×	×	√	×	×	×	×	×	×	×	×	×
	[61]	√	×	×	×	√	×	×	×	×	×	×	×	×	×
	Environment key generation														
	[62]	√	×	×	×	×	√	√	√	√	×	×	×	×	×
	[63]	√	×	×	×	×	√	√	√	√	×	×	×	×	×
	Executing tracking														
	[64]	√	√	×	×	√	√	×	×	√	×	×	×	×	×
	[65]	√	√	×	×	√	√	×	×	√	×	×	×	×	×
	Watermarking														
	[66]	×	√	×	×	×	×	×	×	×	×	×	×	×	×
	[67]	×	√	×	×	×	×	×	×	×	×	×	×	×	×
	Co-signing														
	[68]	√	√	×	×	×	√	×	√	×	×	×	×	×	×
	[69]	√	√	×	×	×	√	×	√	×	×	×	×	×	×
	Separation of privileges														
	[70]	√	√	×	×	√	×	√	√	√	×	×	×	×	×
	[71]	√	√	×	×	√	×	√	√	√	×	×	×	×	×
	Fragmentation-based encryption														
	[72]	×	√	×	√	×	×	×	×	×	×	×	×	×	×
	[73]	√	√	×	×	×	√	×	×	×	×	×	×	×	×

TABLE. IV. SECURITY REQUIREMENTS ACCORDING TO THE CLASS OF SECURITY AGENTS

Class	Distinguished security requirements
protecting mobile agent	Anonymity, Accountability
protecting agent platform	Availability, Non-Repudiation, Verification
Security requirements needed in both classes	
Confidentiality, Integrity, Authentication, Authorization, Accounting, and Assurance	

Based on the conclusions drawn and represented by the points discussed above, we distinguish the security requirements that are individually needed for each of the two classes as well as those needed for both. Table IV separates the security requirements according to the classes. It is worth mentioning that all privacy aspects are needed for the two classes.

VI. PROTECTION GOALS AND ATTACKS

When a mobile agent migrates from an HM to a DM, it might be attacked by the DM. In this section, we define the protection goals that an approach aims to guarantee in protecting the mobile agent class. Then, we explore the potential attacks and measure their impacts on the protection goals based on a maturity-based model.

A. Protection Goals

As mentioned in the introduction section, the mobile agent consists of four main parts, which are the code, the data, the state, and the itinerary. Since the code of the mobile agent and the data are tightly coupled, we refer to them as the first protection goal. The second and third protection goals are the state and the itinerary respectively. To explain how these goals are attacked, we provide an example inspired from a smart city environment, as described below.

In smart cities, ensuring comprehensive safety is an important issue for saving people's lives. Smart warning systems (SWSs) can contribute to achieve this noble goal through alarming the decision makers to take the corresponding steps that ensure avoiding disasters [74, 75, 76]. Fig. 9 illustrates the general concept of SWS.

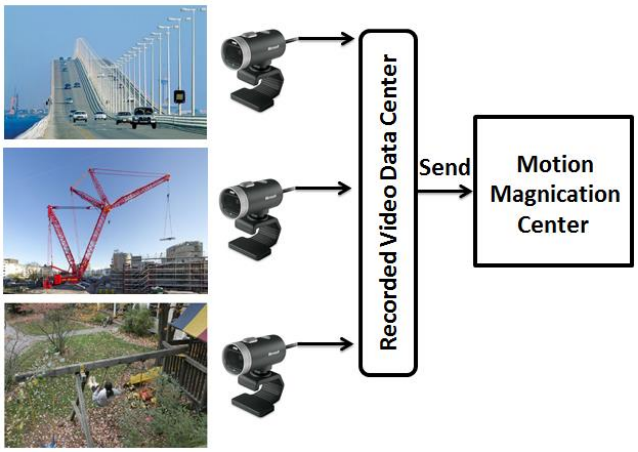


Fig. 9. Smart Warning System.

In Fig. 9, fixed cameras record the motion of different objects in smart cities. The motion magnification centre (MMC) processes the recorded video to enlarge the unseen, abnormal, and critical motions that may cause disasters. Color clustering-based and phase-based video motion processing techniques, which resemble a microscope that amplifies subtle motions in a video sequence allowing visualization of deformations that would otherwise be invisible, can be found in [77, 78].

When agent software technology is employed to build such an SWS, the architecture of the SWS is illustrated in Figure 10.

As shown in Fig. 10, a mobile agent is created at the decision-making centre (the HM), and this mobile agent then migrates to the recording video centre (the DM) to perform a magnification task on the recorded video. After performing the magnification task at the DM, the mobile agent migrates back to the HM carrying the results to be analysed at the decision-making centre for the purpose of avoiding disasters.

As a first scenario, the code and the data of the mobile agent as well as the code of the task can be attacked by the DM, so that a modification can skew the task intended to be performed. As a second scenario, the code and the data are not modified, where the task is performed correctly at the recording video centre, but the results of the task execution are modified. In other words, the mobile agent will carry the wrong results to be analysed at the decision-making centre. As a third and final scenario, when many DMs are involved in the itinerary, the itinerary information of the mobile agent can be changed so that the agent migrates to the wrong next DM. However, in the all three scenarios, the disaster may occur with a high probability.

To avoid any of dangerous scenarios mentioned above, the protection mechanisms should guarantee the protection of the three goals at the same time. In terms of goals' protection, Table V compares the approaches proposed to protect the mobile agent against the DM as an attacker.

As inferred from Table V, the majority of the approaches succeed in protecting the first goal (17 out of 18), while they fail in protecting the second and third goal (1 out of 18 and 2 out of 18, respectively).

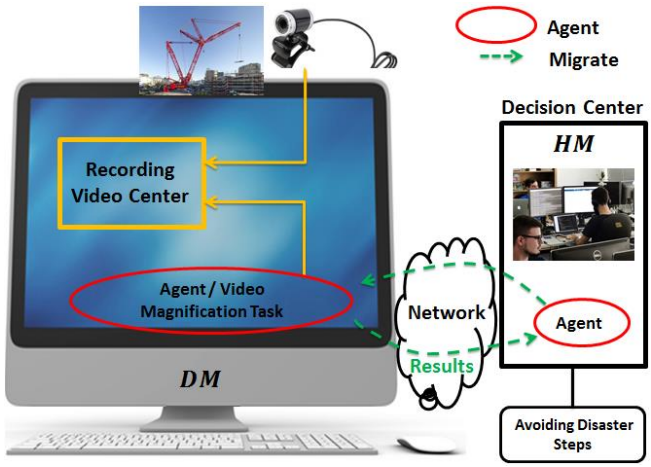


Fig. 10. Agent-based SWS Architecture.

TABLE. V. ACHIEVING PROTECTION GOALS IN PROTECTING MOBILE AGENT APPROACHES

Category	Term / Tech	Protection Goals: Code & Data	State	Itinerary
Protecting mobile Agent	***Collaborative agents***			
	[56]	√	×	√
	[57]	×	×	√
	RPE			
	[58]	√	×	×
	[59]	√	×	×
	Obfuscated code			
	[60]	√	×	×
	[61]	√	√	×
	Environment Key Generation			
	[62]	√	×	×
	[63]	√	×	×
	Execution tracking			
	[64]	√	×	×
	[65]	√	×	×
	Watermarking			
	[66]	√	×	×
	[67]	√	×	×
	Co-signing			
	[68]	√	×	×
	[69]	√	×	×
	Separation of Privileges			
	[70]	√	×	×
	[71]	√	×	×
	Fragmentation based encryption			
	[72]	√	×	×
	[73]	√	×	×

B. Attacks

Exploitation of vulnerabilities is actually considered the spirit of the attacks. Therefore, there is a strong relation between the attacks and vulnerabilities. The protection mechanisms or measures address the control of the vulnerabilities, aiming at mitigating, detecting, or preventing the attacks. Fig. 11 illustrates the relation between vulnerabilities, attacks and protection mechanisms.

For more explanation, Table VI (a and b) elaborates the details of the major security requirements (i.e. CIA) in relation to vulnerabilities, attacks, and protection mechanisms.

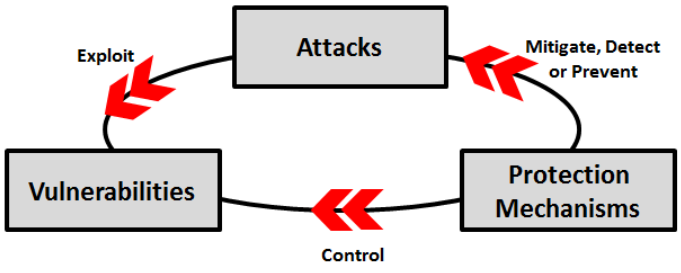


Fig. 11. Relation between Vulnerabilities, Attacks and Protection Mechanisms.

TABLE. VI. (A) CIA IN RELATION TO VULNERABILITIES, ATTACKS, AND PROTECTION MECHANISMS

CIA / Term	1 Confidentiality	2 Integrity		3 Availability	
Vulnerabilities	$v_{1,1}$	$v_{1,2}$		$v_{1,3}$	$v_{1,4}$
Attacks	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	
Protection Mechanisms	$pm_{1,1}$	$pm_{1,2}$	$pm_{1,3}$	$pm_{1,4}$	

(B) DESCRIPTION OF CODES

Code	Description	Code	Description	Code	Description
$v_{1,1}$	Disclosure to unauthorized party.	$a_{1,1}$	Eavesdropping attack	$pm_{1,1}$	Encryption
$v_{1,2}$	Modification by unauthorized party.	$a_{1,2}$	Tampering attack	$pm_{1,2}$	Access control
		$a_{1,3}$	Man-in-the-middle attack	$pm_{1,3}$	Authentication
$v_{1,3}$	Authorized parties are not able to access data.	$a_{1,4}$	DoS attack	$pm_{1,4}$	Hashing and digital signature
$v_{1,4}$	Natural disasters				

C. Classification of Attacks

In the mobile agent-based systems, attacks can be classified based on the nature of the attacks or based on the victim of the attack, as shown in Fig. 12.

According to the nature of the attack, attacks can be passive or active. In passive attacks, the attacker collects some information about the victim to be used later for malicious purposes. Therefore, no updates can affect the resources of the system. In active attacks, the attacker modifies the resources of the system so that there will be direct damage. As a result, active attacks are more dangerous than passive attacks. According to the victim of the attack, malicious agents can attack the operation execution on the platform, and in contrast, malicious platforms can attack the visiting mobile agents. Attacking the visiting mobile agents by platforms is more dangerous than attacking the platform by agents, which is because the platform has full control of the execution of the visiting agents within its operational environment. Table VII summarizes the most common possible attacks that agent-based systems suffer from, as well as their types.

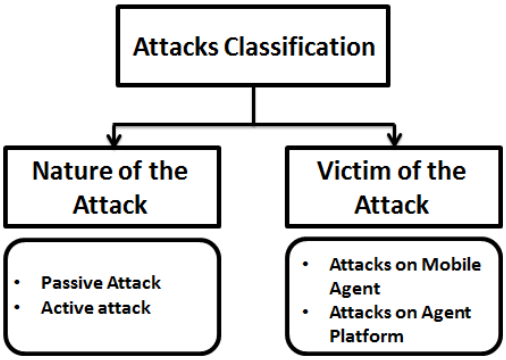


Fig. 12. Classification of Attacks in Agent-based Systems.

TABLE. VII. POSSIBLE ATTACKS

Victim of attack	Possible attacks	Nature of attack	
		Active	*Passive*
Mobile agent	1- DoS attack by the host of the agent	√	
	3- Eavesdropping on an agent's activities		√
	3- Blocking attack by the host	√	
	4- Modification of an agent by the host	√	
	5- Multiple colluded attack by hosts	√	
Agent platform	1- DoS attack with overmuch requests or exhausting the platform's memory or resources	√	
	2- Unauthorized access attack for: * shutting down platform * modifying policy file * performing any malicious activity	√	

D. Overview of Attacks

For the attacks that target the agent platform by a malicious agent, Fig. 13 and Fig. 14 illustrate the mechanisms by which the DoS and unauthorized access attacks are performed.

As shown in Fig. 13, a malicious agent migrates from an HM to a DM, asking to execute infinite requests of the same mission, so that the DM goes through an infinite loop of execution, thereby resulting in exhausting the resources of the host machine. In this case, if any other agent that exists in the DM asks for execution of its own mission, it is forced to wait forever, due to the allocation of the DM's resources for the malicious agent [79].

In Fig. 14, a malicious agent gains unauthorized access to a DM by exploiting some gaps in the system. After accessing the DM, the malware carried by the agent is then executed to damage some critical system files [80].

For the attacks that target the agent itself by a malicious DM, the DoS attack has a special case, as shown in Fig. 15.

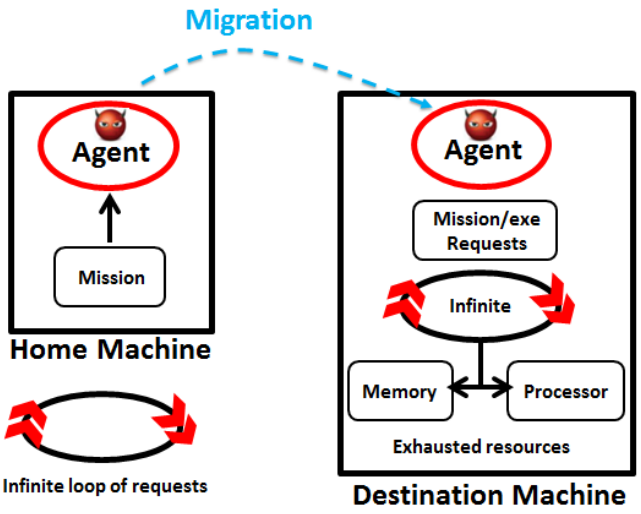


Fig. 13. Concept of the DoS Attack.

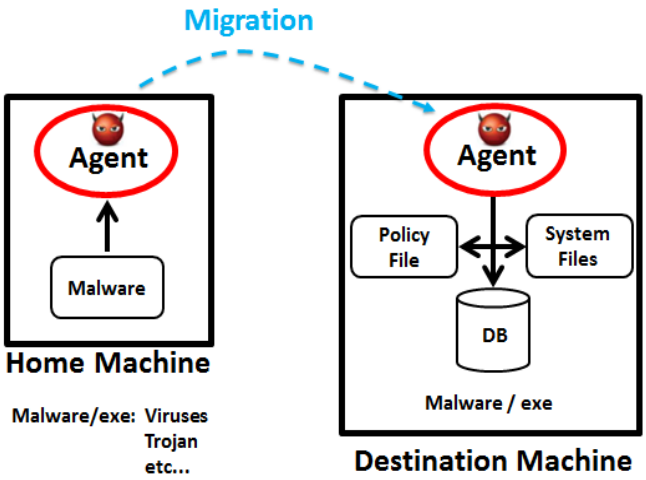


Fig. 14. Concept of the unauthorized Access Attack.

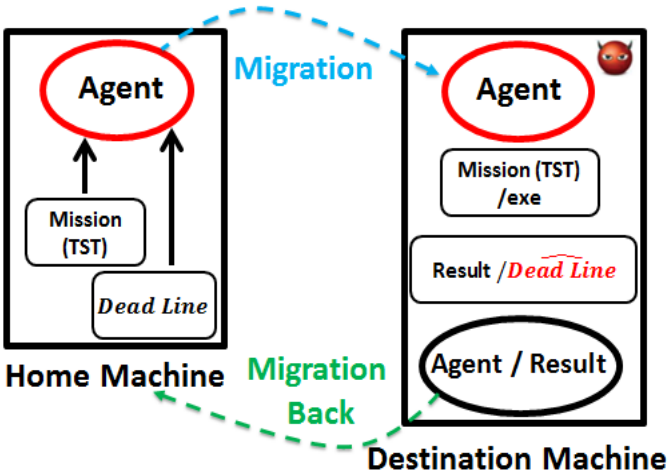


Fig. 15. Special Case of the DoS Attack.

In the DoS attack, the mobile agent carries a mission (time sensitive task TST), for which its execution time is restricted by a deadline ($T_{Dead\ Line}$). After the mobile agent migrates to the DM, the TST is executed there. The malicious DM deliberately delays (or lengthens) the execution time of the TST, so that it exceeds the predefined $T_{Dead\ Line}$. It is worth mentioning that even when the malicious DM does not modify the result of the TST, the result will be invalid and unused when it is received by the HM [81, 82].

Eavesdropping on an agent is applied by the malicious DM by monitoring and recording the activities of the agent during the time in which the agent executes its mission [83]. In the worst case, the eavesdropping attack can be turned into a blocking attack, where the DM completely prevents the agent from execution after a short period of monitoring [84].

Fig. 16 shows how the modification attack is applied on the visiting mobile agent by the DM. The DM does not monitor or block the agent, such that it is allowed to execute smoothly, but after generating the results of the execution, the DM tampers or changes the results. In other words, the mobile agent migrates back to the HM carrying wrong results [85].

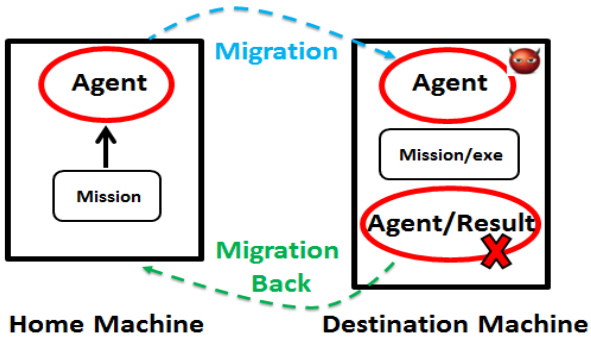


Fig. 16. Concept of the Modification Attack.

The modification attack becomes more dangerous when two or more DMs collude together to modify the results of the execution. This kind of attack is called the multiple colluded attack, as illustrated in Fig. 17.

In the multiple colluded attack, a series of (n) DMs ($DM_1, DM_2, \ldots DM_{n-1}, DM_n$) collude each other for a common malicious purpose, which is tricking the HM. Tricking the HM is achieved in such a way that: (1) two DMs, or more, modify the result generated after execution of the mission of the visiting mobile agent; and (2) all malicious DMs that modify the result use the same modification process [50]. Upon enacting this, the success of the multiple colluded attack can be adjusted by the following two conditions:

1) The original result ($result_i$), generated at ($DM_i | 0 < i \leq n$), is modified to be ($\widehat{result}_i$).

2) At the HM, all the received results are modified, so that: $\widehat{result}_1 = \widehat{result}_2 = \cdots = \widehat{result}_{n-1} = \widehat{result}_n$

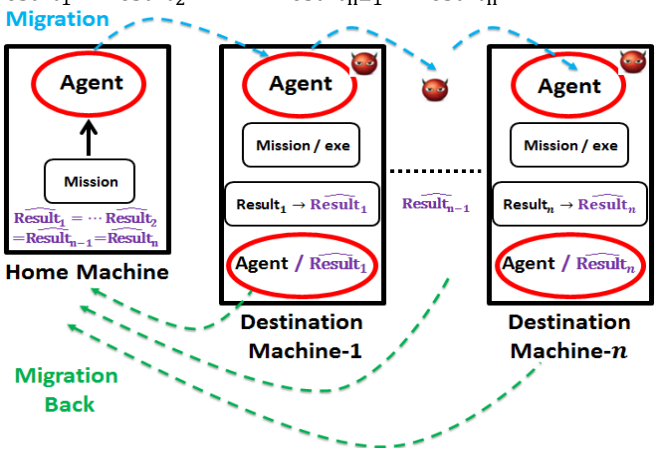


Fig. 17. Concept of the Multiple Colluded Attack.

E. Maturity Model

To show the negative impact of the attacks, we propose a statistical model called the maturity model. The maturity model deals with the protection goals (i.e., code, status, and itinerary) as affected aspects of attacks. Since the protection goals are limited to the class of protecting the mobile agent compared to unlimited protection goals in the class of protecting the agent platform (i.e., any part of the system in the DM's platform can be a victim of attack), we deal only with those attacks that target the mobile agent as a victim (Table VII). Among the attacks that target the mobile agent as a victim (in Table VII), we consider only the DoS, modification, and multiple colluded attacks since they are considered as advanced attacks. Upon this consideration, in the maturity model, the DoS, modification, and multiple colluded attacks are considered as main criteria factors, while the protection goals are considered as affected aspects. All approaches contained in Table V above are evaluated. Our evaluation relies on three options to measure the negative impact of the criteria factors. Table VIII provides a description of the three used options.

TABLE. VIII. OPTIONS OF MEASUREMENT

Option	Description
√	When the factor has high negative impact.
×	When the factor has a low negative impact.
P	When the factor has a partially negative impact.

1) Analysis and discussion: Table IX below can be read horizontally or vertically, as illustrated in Fig. 18.

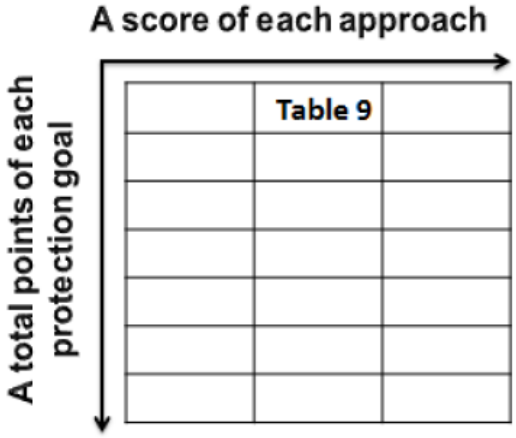


Fig. 18. Horizontal and Vertical Reading of Table IX.

TABLE. IX. EFFECT OF THE DOS ATTACK

Class	Technique / Term	Protection Goals			Sub Totals		
		Code & Data	*State*	*Itinerary*	√	×	*P*
Protecting mobile Agent	***Collaborative agents***		Maturity: 2				
	[56]	P	√	×	1	1	1
	[57]	×	√	×	1	2	0
	RPE		Maturity: 2				
	[58]	P	√	×	1	1	1
	[59]	P	√	×	1	1	1
	Obfuscated code		Maturity: 1				
	[60]	P	√	×	1	1	1
	[61]	P	P	×	0	1	2
	Environment Key Generation		Maturity: 2				
	[62]	×	√	×	1	2	0
	[63]	×	√	×	1	2	0
	Execution tracking		Maturity: 2				
	[64]	P	√	×	1	1	1
	[65]	P	√	×	1	1	1
	Watermarking		Maturity: 3				
	[66]	√	√	×	2	1	0
	[67]	√	P	×	1	1	1
	Co-signing		Maturity: 0				
	[68]	P	P	×	0	1	2
	[69]	P	P	×	0	1	2
	Separation of Privileges		Maturity: 2				
	[70]	P	√	×	1	1	1
	[71]	P	√	×	1	1	1
	Fragmentation-based encryption		Maturity: 0				
	[72]	P	P	×	0	1	2
	[73]	P	P	×	0	1	2
Sub Totals:							
√	High negative impact	2	12	0	14		
×	Low negative impact	3	0	18	21		
P	Partial negative impact	13	6	0	19		
Total:					54		

If Table IX is read horizontally, then the numbers on the table represent the total points that each approach has obtained from all of the protection goals for each one of the three options. Each option has a score that varies in the range of [0, 1, 2, 3]. For instance, the corresponding numbers of the collaborative agents' technique are 2, 3, and 1 for the √, ×, and P options respectively. This in turn means that the DoS attack has a moderate impact on the approaches proposed under this technique because the score of the √ option equals 2. Thus, the maturity of the collaborative agents' technique is moderate under the threat of the DoS attack. A reasonable justification is that this technique was originally designed to protect the code itself, where assistant agents contribute to prevent the illegal redundancy of the same request (or the mission under execution). RPE, environment key generation, execution tracking, and separate of privileges techniques have the same level of maturity as the collaborative agents' technique. The maturity of the watermarking-based technique under the threat of the DoS attack is low because the score of the √ option equals 3, which is because the objective of this technique is to detect any modification in the code, not to prevent redundancy. The maturity of the obfuscated code-based technique under the threat of the DoS attack is high because the score of the √ option equals 1. The score is 1 because the obfuscation of the

code prevents the attacker from extracting the real code so that it can be redundant. For the co-signing and fragmentation-based encryption techniques, the maturity level under the threat of the DoS attack is very high since the score of the √ option in each one equals 0. The score is 0 because the code is highly protected against being decrypted, and then against being exploited for redundancy. Within four groups, Table X ranks the previous techniques according to their maturity levels.

If Table IX is read vertically, then the numbers represent the total points that each protection goal has obtained for each one of the three options and is related to all of the approaches provided in protecting the mobile agent class. From the numbers that appear in Table IX, it can be noticed that the total number of points that the (√) option achieved is 14 points. These points distribute over the (code & data, state, and itinerary) protection goals with (2, 12, and 0) values respectively. For the (×) option, the protection goals achieved (3, 0, and 18) values from the sub-total points, with a total of 21. The corresponding values related to the protection goals for the (P) option are (13, 6, and 0) from the sub-total points, with a total of 19. In terms of percentages, Fig. 19(a) above shows the negative impact of the DoS attack's threat on the protection goals of all techniques (i.e., overall maturity against the DoS attack).

In Fig. 19(a), the high negative impact option (√) has the lowest percentage (0.25), while the low negative impact option (×) has the highest percentage. This in turn reflects a good maturity of the security approaches against the DoS attack. The reason behind this is that most of the approaches in all techniques are designed to protect the code of the agent.

Regarding modification and multiple colluded attacks, we rebuilt Table IX, scanned it vertically, and determined the percentages, as shown in Fig. 19(b), (c) above, respectively.

According to Fig. 19(b) and (c), most of the approaches suffer from the modification attack, with the percentage equal to 0.73 for the high negative impact option, which indicates a low overall maturity level. Compared to the modification attack, the security approaches have very poor maturity against the multiple colluded attack, with the percentage equal to 0.92 for the high negative impact option. The reason behind this is that both the state (which contains the results of the mission execution) and the code can be modified by a series of malicious DMs during the itinerary of the mobile agent.

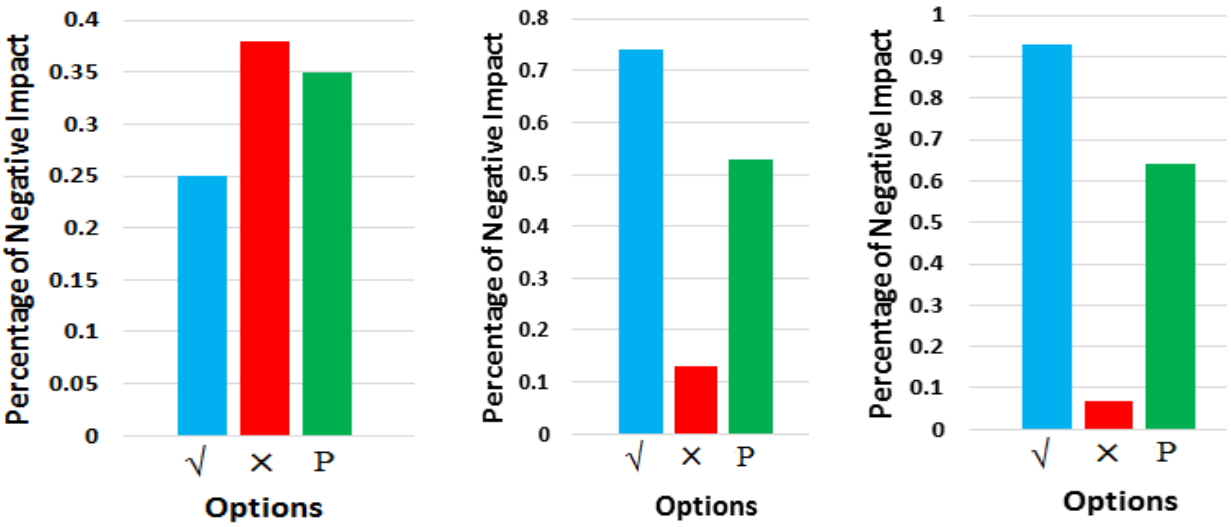
Based on the analysis derived from Fig. 19, Table XI ranks the attacks according to their danger on the visiting mobile agents.

TABLE. X. RANKING TECHNIQUES ACCORDING TO MATURITY LEVEL

Group	Techniques	Score of √ option	Maturity level
G1	Co-signing, fragmentation-based encryption	0	Very high
G2	Obfuscated code	1	High
G3	Collaborative agents, RPE, environment key generation, execution tracking, separation of privileges	2	Moderate
G4	Watermarking	3	Low

TABLE. XI. DANGER-BASED RANKING ATTACKS

Attack	Percentage of high negative impact option (√)	Ranking
Multiple colluded attack	0.92	First
Modification attack	0.73	Second
DoS attack	0.25	Third



(a) Overall Negative Impact of DoS attack (b) Maturity Against Modification attack (c) Maturity Against Multiple Colluded attack.

Fig. 19. Overall Negative Impact of Modification and Multiple Colluded Attacks.

VII. Evaluation of Security of Agent-Based Systems

Designing a secure agent-based system requires defining a threat model at the beginning. The clear threat model contains four parts, as shown in Fig. 20.

The first part of the threat model is the attacker. In agent-based systems, the attacker can be the mobile agent, the destination machine, or any malicious party in the system such as the man-in-the-middle. After deciding who the attacker is, it determines his objective, which is the second part on the threat model. The objective of the attacker can be attacking one or more of the protection goals if the victim is the mobile agent or applying any malicious activity if the victim is the agent platform. Based on the determined objective, the type of the attack is defined to be active or passive. Finally, the capability of the attacker is listed in the context of the attacks that the attacker can apply, such as eavesdropping, modification, DoS, or multiple colluded attacks, etc.

After defining the threat model, the agent-based system is built so that defenses (or security controls) against the capability of the attacker are implemented. After building the system, it undergoes a validation process to ensure that the security controls are able to detect or prevent the attacks. Finally, to measure the efficiency of the security controls, security metrics-based evaluation must be performed. In this context, we explore the security metrics used in agent-based systems. We classify the security metrics into three main categories, as shown in Fig. 21.

For the first group, the number of security requirements that are achieved is used as a metric to evaluate the proposed agent-based system. Therefore, the strength of the system is linked to satisfying a higher number of security requirements. An available tool, called Scyther [86], can be employed in this group to check if a certain security requirement is satisfied or not.

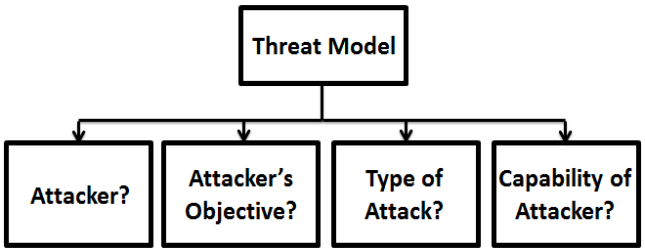


Fig. 20. Parts of the Threat Model.

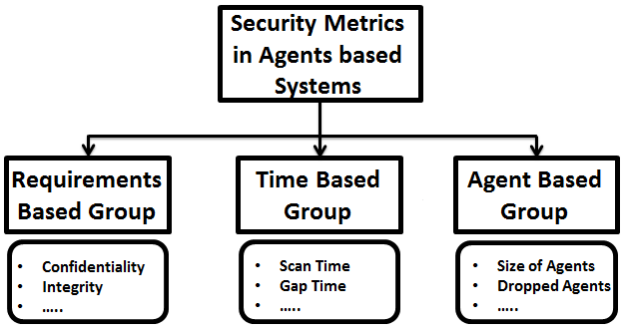


Fig. 21. Classification of Security Metrics.

For the second group, time is mainly employed in different forms as a metric. Such forms are: (1) the time needed for scanning the code of the agent; (2) the time needed for encryption and decryption; (3) the time needed for calculation of the hash of the agent's code; and (4) the time gap, which is the time consumed between the attack detection and the actual update that is applied to the system.

For the third group, the size of the agent can be used as a metric as well as the events that occur in the system and are related to the agent, such as the number of agents that are dropped by the platform, and the number of agents that are attacked in time units.

VIII. Challenges and Research Questions

In this section, we provide the challenges encountered by the corresponding research questions (five questions) according to the section presented in this work. In addition, we present an additional two research questions as inferred ones.

1) According to Section III (Cyber Security requirements in the lifecycle of mobile agent), requirements related to security and those related to privacy are the challenges. The two kinds of requirements should be achieved in agent-based systems to state that we have a comprehensive secure system in terms of cyber security. However, researchers distinguish between security and privacy and rank security at the top compared to privacy. Therefore, how to satisfy all the security requirements simultaneously in an agent-based system is the first research question.

2) According to Section IV (Classification of security techniques in mobile agents), the visiting mobile agent my damage the destination machine, and at the same time, the visited destination machine may attack the incoming agent. This leads to the challenge that can be represented by the second research question: how to ensure that the visiting mobile agent and the host machine do not attack each other so that each party operates in a completely secure manner.

3) According to section VI (Protection goals and attacks), two research questions are motivated as follows:

- Since one of the most important benefits of mobile agents is reducing the network overhead as well as solving the transmission challenge problem, protecting the state (which carries the results of execution back to the home machine) is a major challenge. Therefore, the research question related to how to guarantee the integrity of the results is a critical one to be answered.

- Regarding attacks, the major concern occurs when the destination machine tries to apply advanced attacks on the visiting mobile agent, such as modification, DoS, and multiple colluded attacks. Therefore, how to build strong protection mechanisms that ensure high resistance against such attacks is the corresponding research question.

4) According to Section VII (evaluation of security of agent-based systems), ensuring efficient and standard security metrics related to the protection mechanisms (i.e., out of the

three groups mentioned in Fig. 21) is a challenge. Therefore, how to quantify the security of agent-based systems by an efficient mathematical model is an important aspect for the evaluation process. This will be a strong point with respect to comparing different protection mechanisms.

5) It is argued that the need of an effective protection mechanism is mandatory. However, how to equip the agent with a protection mechanism that provides a self-protection feature is another research question.

6) Since the destination machine has full control over the visiting mobile agent, it is important from the security point of view to isolate communications with the destination machine. In other words, during the performance of the agent's mission within the operational environment of the destination machine, how to endow the agent with a self-communication feature is an overarching research question.

IX. Conclusion

Compared to other software technologies, agent-based software technology presents itself as an effective solution for many problems in distributed systems, such as network overhead and transmission challenge. However, the security issue is a main factor that contributes to limitations of the benefits of agent-based software technology as well as its applications. The main reason behind this issue is that the agents can be attacked by the destination machines where they perform the missions, or the visiting agents can perform malicious activities on the host machine. Moreover, advanced attacks such as DoS, modification, and multiple colluded attacks can exacerbate the security problem. Based on the attacker (the visiting mobile agent and the destination or host machine), we review different techniques used to ensure the security in agent-based systems, critique them, and compare them according to well-defined cyber security requirements (in both the security and privacy aspects). Based on protection goals (code and data, state, and itinerary of the mobile agent), a maturity model is employed to analyse the security techniques as well as rank the strength of the attacks. Finally, seven research questions are provided in the research field of agent security that should be answered to ensure comprehensive security in agent-based systems.

References

[1] Padgham, Lin, and John Thangarajah. "Agent Oriented Software Engineering: Why and How." VNU Journal of Science: Natural Sciences and Technology 27.3 (2016).

[2] Qiu, Linrun, and Kangshun Li. "The Research of Intelligent Agent System Architecture Based on Cloud Computing." Computational Intelligence and Security (CIS), 2016 12th International Conference on. IEEE, 2016.

[3] [Caglayan, Alper, and Colin Harrison. "Agent sourcebook." (2011).

[4] Satoh, Ichiro. "Building reusable mobile agents for network management." IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 33.3 (2003): 350-357.

[5] Bieszczad, Andrzej, Bernard Pagurek, and Tony White. "Mobile agents for network management." IEEE Communications Surveys 1.1 (1998): 2-9.

[6] Bergenti, Federico, Eleonora Iotti, and Agostino Poggi. "Core features of an agent-oriented domain-specific language for JADE agents." Trends in Practical Applications of Scalable Multi-Agent Systems, the PAAMS Collection. Springer, Cham, 2016. 213-224.

[7] Urra, Oscar, et al. "Mobile agents and mobile devices: Friendship or difficult relationship?." (2009). J. Phys. Agents 3, 2 (2009), 27−37.

[8] Mobil Agent Computing website, online, available: https://www.cis.upenn.edu/~bcpierce/courses/629/papers/Concordia-WhitePaper.html , (2018), 6th October.

[9] Jade website, online, available: http://jade.tilab.com/ , (2018), 6th October.

[10] Binu A website, online, available: http://csr.cusat.ac.in/people/binua /blog/ibm-aglets-workbench-installation-agent-programming,(2018), 6th October.

[11] Wu, Bing, et al. "A survey of attacks and countermeasures in mobile ad hoc networks." Wireless network security. Springer, Boston, MA, 2007. 103-135.

[12] Karnik, Neeran M., and Anand R. Tripathi. "A security architecture for mobile agents in Ajanta." Distributed Computing Systems, 2000. Proceedings. 20th International Conference on. IEEE, 2000.

[13] Bakre, Ajay, and B. R. Badrinath. "M-RPC: A remote procedure call service for mobile clients." Proceedings of the 1st annual international conference on Mobile computing and networking. ACM, 1995.

[14] Stamos, James W., and David K. Gifford. "Implementing remote evaluation." IEEE Transactions on Software Engineering 16.7 (1990): 710-722.

[15] Carzaniga, Antonio, Gian Pietro Picco, and Giovanni Vigna. "Designing distributed applications with mobile code paradigms." Proceedings of the 19th international conference on Software engineering. ACM, 1997.

[16] Ali, Anwaar, et al. "Big data for development: applications and techniques." Big Data Analytics 1.1 (2016): 2.

[17] Ramírez-Gallego, Sergio, et al. "Big data: tutorial and guidelines on information and process fusion for analytics algorithms with MapReduce." Information Fusion 42 (2018): 51-61.

[18] Boubiche, Sabrina, et al. "Big Data Challenges and Data Aggregation Strategies in Wireless Sensor Networks." IEEE Access 6 (2018): 20558-20571.

[19] Zhu, Li, et al. "Big Data Analytics in Intelligent Transportation Systems: A Survey." IEEE Transactions on Intelligent Transportation Systems (2018).

[20] Kavak, Hamdi, et al. "Big data, agents, and machine learning: towards a data-driven agent-based modeling approach." Proceedings of the Annual Simulation Symposium. Society for Computer Simulation International, 2018.

[21] Karami, Mahtab, and Ali HOSSEINI SHAHMIRZADI. "Applying Agent-based Technologies in Complex Healthcare Environment." Iranian journal of public health 47.3 (2018): 458.

[22] Kaeri, Yuki, et al. "Agent-Based System Architecture Supporting Remote Collaboration via an Internet of Multimedia Things Approach." IEEE Access 6 (2018): 17067-17079.

[23] Sun, Weiwei, et al. "An air index for spatial query processing in road networks." IEEE Transactions on Knowledge and Data Engineering 27.2 (2015): 382-395.

[24] Alrahhal, Mohamad Shady, Maher Khemekhem, and Kamal Jambi. "Achieving load balancing between privacy protection level and power consumption in location based services." (2018).

[25] Lange, Danny B., and Mitsuru Oshima. "Seven good reasons for mobile agents." Communications of the ACM 42.3 (1999): 88-89.

[26] Chaabouni, Taha, and Maher Khemakhem. "Resource Management Based on Agent Technology in Cloud Computing." Advances in Information Technology for the Holy Quran and Its Sciences (32519), 2013 Taibah University International Conference on. IEEE, 2013.

[27] Arfat, Yasir, and Fathy Elbouraey Eassa. "A Survey on Fault Tolerant Multi Agent System." IJ Inf. Technol. Comput. Sci 9 (2016): 39-48.

[28] Madkour, Mohamed A., et al. "Mobile Agent Framework for Distributed Network Performance Management." International Journal of Computer Applications 88.5 (2014).

[29] Imran, Muhammad, Fathy Eassa, and Kamal Jambi. "Using Agent Technology for Security Testing of WEB Based Applications." SEDE– 2015: 3-10.

[30] Alrahhal, Mohamad Shady, Maher Khemakhem, and Kamal Jambi. "Agent-Based System for Efficient kNN Query Processing with Comprehensive Privacy Protection." International Journal Of Advanced Computer Science And Applications 9.1 (2018): 52-66.

[31] Wernke, Marius, et al. "A classification of location privacy attacks and approaches." Personal and ubiquitous computing18.1 (2014): 163-175.

[32] Alrahhal, Mohamad Shady, Maher Khemakhem, and Kamal Jambi. "A Survey on Privacy of Location-Based Services: Classification, Inference Attacks, And Challenges." Journal of Theoretical & Applied Information Technology 95.24 (2017).

[33] Alrahhal, Mohamad Shady, et al. "AES-Route Server Model for Location based Services in Road Networks." International Journal Of Advanced Computer Science And Applications 8.8 (2017): 361-368.

[34] Kharaji, Morteza Yousefi, and Fatemeh Salehi Rizi. "A fast survey focused on methods for classifying anonymity requirements." International Journal of Computer Science and Information Security 12.4 (2014): 59.

[35] Deng, Mina. "Privacy Preserving Content Protection (Privacy behoud content protection)." (2010).

[36] Wahbe, Robert, et al. "Efficient software-based fault isolation." ACM SIGOPS Operating Systems Review. Vol. 27. No. 5. ACM, 1994.

[37] Van't Noordende, Guido, Frances MT Brazier, and Andrew S. Tanenbaum. "A security framework for a mobile agent system." Proceedings of the 2nd International Workshop on Security in Mobile Multiagent Systems (SEMAS 2002), associated with AAMAS-2002, Bologna, Italy. 2002.

[38] Rights, Retains Full. "Secure Coding. Practical steps to defend your web apps." (2007).

[39] Malik, Najmus Saqib, and Albert Treytl. "Optimizing Security Computation Cost for Mobile Agent Platforms." Industrial Informatics, 2007 5th IEEE International Conference on. Vol. 1. IEEE, 2007.

[40] Cooper, David, et al. Security Considerations for Code Signing. No. OTHER-. 2018.

[41] Appel, Andrew W., and David McAllester. "An indexed model of recursive types for foundational proof-carrying code." ACM Transactions on Programming Languages and Systems (TOPLAS) 23.5 (2001): 657-683.

[42] Sekar, R., et al. "Model-carrying code: a practical approach for safe execution of untrusted applications." ACM SIGOPS Operating Systems Review. Vol. 37. No. 5. ACM, 2003.

[43] Kim, Hyong-Soon, and Eunyoung Lee. "Verifying Code toward Trustworthy Software." Journal of Information Processing Systems 14.2 (2018).

[44] Borselius, Niklas. "Mobile agent security." Electronics & Communication Engineering Journal 14.5 (2002): 211-218.

[45] Chess, David, et al. "Itinerant agents for mobile computing." IEEE Personal Communications 2.5 (1995): 34-49.

[46] Ordille, Joann J. "When agents roam, who can you trust?." Emerging Technologies and Applications in Communications, 1996. Proceedings., First Annual Conference on. IEEE, 1996.

[47] Guan, Sheng-Uei, Tianhan Wang, and Sim-Heng Ong. "Migration control for mobile agents based on passport and visa." Future Generation Computer Systems 19.2 (2003): 173-186.

[48] Marikkannu, P., R. Murugesan, and T. Purusothaman. "AFDB security protocol against colluded truncation attack in free roaming mobile agent environment." Recent Trends in Information Technology (ICRTIT), 2011 International Conference on. IEEE, 2011.

[49] Hefeeda, Mohamed, and Bharat Bhargava. "On mobile code security." Purdue University (Oct. 2001) (2001).

[50] Venkatesan, S., C. Chellappan, and P. Dhavachelvan. "Performance analysis of mobile agent failure recovery in e-service applications." Computer Standards & Interfaces 32.1-2 (2010): 38-43.

[51] Venkatesan, S., et al. "Artificial immune system based mobile agent platform protection." Computer Standards & Interfaces35.4 (2013): 365-373.

[52] Tsiligiridis, Theodore A. "Security for mobile agents: Privileges and state appraisal mechanism." Neural Parallel And Scientific Computatiions 12.2 (2004): 153-162.

[53] Farmer, William M., Joshua D. Guttman, and Vipin Swarup. "Security for mobile agents: Authentication and state appraisal." European Symposium on Research in Computer Security. Springer, Berlin, Heidelberg, 1996.

[54] Bagga, Pallavi, Rahul Hans, and Vipul Sharma. "N-grams Based Supervised Machine Learning Model for Mobile Agent Platform Protection against Unknown Malicious Mobile Agents." International Journal of Interactive Multimedia & Artificial Intelligence 4.6 (2017).

[55] Bagga, Pallavi, Rahul Hans, and Vipul Sharma. "A Biological Immune System (BIS) inspired Mobile Agent Platform (MAP) security architecture." Expert Systems with Applications 72 (2017): 269-282.

[56] Roth, Volker. "Mutual protection of co-operating agents." Secure Internet Programming. Springer, Berlin, Heidelberg, 1999. 275-285.

[57] Madkour, A. M., et al. "Securing mobile-agent-based systems against malicious hosts." World Applied Sciences Journal 29.2 (2014): 287-297.

[58] Yee, Bennet S. "A sanctuary for mobile agents." Secure Internet Programming. Springer, Berlin, Heidelberg, 1999. 261-273.

[59] Young, Adam, and Moti Yung. "Sliding encryption: a cryptographic tool for mobile agents." International Workshop on Fast Software Encryption. Springer, Berlin, Heidelberg, 1997.

[60] Hohl, Fritz. "Time limited blackbox security: Protecting mobile agents from malicious hosts." Mobile agents and security. Springer, Berlin, Heidelberg, 1998. 92-113.

[61] Badger, Lee, et al. "Self-protecting mobile agents obfuscation techniques evaluation report." Network Associates Laboratories, Report (2002): 01-036.

[62] Riordan, James, and Bruce Schneier. "Environmental key generation towards clueless agents." Mobile agents and security. Springer, Berlin, Heidelberg, 1998. 15-24.

[63] Tschudin, Christian. "Apoptosis—The programmed death of distributed services." Secure Internet Programming. Springer, Berlin, Heidelberg, 1999. 253-260.

[64] Tan, Hock Kim, and Luc Moreau. "Extending execution tracing for mobile code security." Proceedings of Second International Workshop on Security of Mobile MultiAgent Systems (SEMAS'2002). 2002.

[65] Tan, Hock Kim, and Luc Moreau. "Certificates for mobile code security." Proceedings of the 2002 ACM symposium on Applied computing. ACM, 2002.

[66] Esparza, Oscar, et al. "Mobile agent watermarking and fingerprinting: tracing malicious hosts." International Conference on Database and Expert Systems Applications. Springer, Berlin, Heidelberg, 2003.

[67] Esparza, Oscar, et al. "Punishing manipulation attacks in mobile agent systems." Global Telecommunications Conference, 2004. GLOBECOM'04. IEEE. Vol. 4. IEEE, 2004.

[68] Cheng, Jeff SL, and Victor K. Wei. "Defenses against the truncation of computation results of free-roaming agents." International Conference on Information and Communications Security. Springer, Berlin, Heidelberg, 2002.

[69] Linna, Fan, and Liu Jun. "A free-roaming mobile agent security protocol against colluded truncation attack without trusted third party." Business Management and Electronic Information (BMEI), 2011 International Conference on. Vol. 2. IEEE, 2011.

[70] Al-Jaljouli, Raja, and Jemal H. Abawajy. "Secure mobile agent-based e-negotiation for on-line trading." Signal Processing and Information Technology, 2007 IEEE International Symposium on. IEEE, 2007.

[71] Traut, Eric, et al. "Protection agents and privilege modes." U.S. Patent No. 8,380,987. 19 Feb. 2013.

[72] Srivastava, Shashank, and G. C. Nandi. "Fragmentation based encryption approach for self protected mobile agent." Journal of King Saud University-Computer and Information Sciences 26.1 (2014): 131-142.

[73] El Rhazi, Abdelmorhit, Samuel Pierre, and Hanifa Boucheneb. "A secure protocol based on a sedentary agent for mobile agent environments." Journal of Computer Science 3.1 (2007): 35-42.

[74] Arepalli, Abhishek, S. Srinivasa Rao, and P. Jagadeeshwara Rao. "A Spatial Disaster Management Framework for Smart Cities—A Case." Proceedings of International Conference on Remote Sensing for Disaster Management: Issues and Challenges in Disaster Management. Springer, 2018.

[75] Hartama, D., et al. "Smart City: Utilization of IT resources to encounter natural disaster." Journal of Physics: Conference Series. Vol. 890. No. 1. IOP Publishing, 2017.

[76] Abid, A., A. Kachouri, and A. Mahfoudhi. "Data analysis and outlier detection in smart city." Smart, Monitored and Controlled Cities (SM2C), 2017 International Conference on. IEEE, 2017.

[77] Liu, Ce, et al. "Motion magnification." ACM transactions on graphics (TOG). Vol. 24. No. 3. ACM, 2005.

[78] Wadhwa, Neal, et al. "Phase-based video motion processing." ACM Transactions on Graphics (TOG) 32.4 (2013): 80.

[79] Mittal, Praveen, and Manas Kumar Mishra. "Trust and Reputation-Based Model to Prevent Denial-of-Service Attacks in Mobile Agent System." Towards Extensible and Adaptable Methods in Computing. Springer, Singapore, 2018. 297-307.

[80] Samet, Donies, Farah Barika Ktata, and Khaled Ghedira. "Securing Mobile Agents, Stationary Agents and Places in Mobile Agents Systems." KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications. Springer, Cham, 2018.

[81] Wood, Anthony D., and John A. Stankovic. "A taxonomy for denial-of-service attacks in wireless sensor networks." Handbook of sensor networks: compact wireless and wired sensing systems (2004): 739-763.

[82] Greenberg, Michael S., Jennifer C. Byington, and David G. Harper. "Mobile agents and security." IEEE Communications magazine 36.7 (1998): 76-85.

[83] Prem, M. Vigilson, and S. Swamynathan. "Securing mobile agent and its platform from passive attack of malicious mobile agents." IEEE-International Conference On Advances In Engineering, Science And Management (ICAESM-2012). IEEE, 2012.

[84] Singh, Rajwinder, and Mayank Dave. "Rescuing data of mobile agents blocked by malicious hosts in e-service applications." 2011 International Conference on Multimedia, Signal Processing and Communication Technologies. IEEE, 2011.

[85] Mitrovic, Nikola, and Unai Arronategui Arribalzaga. Mobile Agent security using Proxy-agents and Trusted domains. No. INPRO--2009-035. 2002.

[86] Cremers, Cas JF. "The scyther tool: Verification, falsification, and analysis of security protocols." International Conference on Computer Aided Verification. Springer, Berlin, Heidelberg, 2008.