International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 04 | Apr 2020 www.irjet.net p-ISSN: 2395-0072

Achieving Self-Protection and Self-Communication Features for Security of Agent-based Systems

Bandar Alluhaybi[1], Mohamad Shady Alrahhal[2], Ahmed Alzhrani[3], Vijey Thayananthan[4]

[1,2,3,4]King Abdul-Aziz University, Faculty of Computing and Information Technology, Department of Computer Science, Jeddah, Saudi Arabia

---

Abstract - *Recently, agent-based software technology has received significant attention from research committees due to its vital role in performing tasks remotely in distributed systems. However, security is a major concern in this technology. When a mobile agent carries a task to be performed remotely, the visiting mobile agent may be attacked by the visited destination machine. On the opposite side, the visited destination machine may be attacked by the visiting mobile agent. This conflict in ensuring security at both the mobile agent side and the destination machine side is the key problem addressed in this work. In this work, we present a Module-based Security System (MbSS) that ensures the security of the visiting mobile agent against the destination machine. In addition, it ensures that the visiting mobile agent does not exceed its privileges to damage the destination machine. MbSS adds a kind of isolation to the mobile agent when performing a task within the space of the destination machine. This isolation is represented by self-decryption and self-communication features. Compared to similar security systems, the proposed MbSS shows better performance in terms of security level and total execution time.*

Key Words: mobile agent, destination machine, encryption, decryption, module, security requirements.

1. INTRODUCTION

Importance of agents. Agent-based systems play a vital role in distributed systems to perform tasks remotely. The agent is a mobile code that can perform tasks on behalf of a network user. The importance of mobile agents comes from their contribution to many research fields, such as network management tasks and information management systems [1, 2]. To perform a task, the owner of the agent first creates the agent. The agent migrates from the Home Machine (HM) to a Destination Machine (DM) through an itinerary predefined by the owner of the agent. The itinerary may include more than one destination machine [3]. Figure 1 illustrates the general scenario of performing a carried task by a mobile agent.

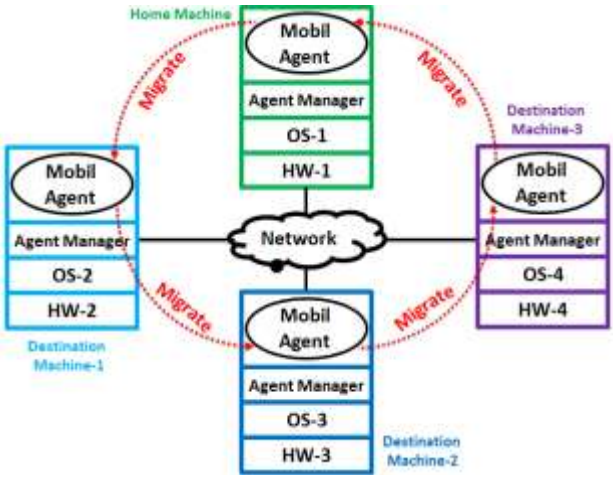

Fig -1: General scenario of performing a task by a mobile agent (migration).

Statement of the problem. The scenario shown in Figure 1 is completely unsecure. On the one hand, the visiting mobile agent may be attacked by any DM, and the problem is accentuated by the mobile agent executing within the space of the visited DM. On the other hand, the visited destination machine tries to ensure that the visiting mobile agent is benign (i.e., it will cause no damage such as updating the security profile or accessing sensitive data). This trade-off forms the key problem, as shown in Figure 2.

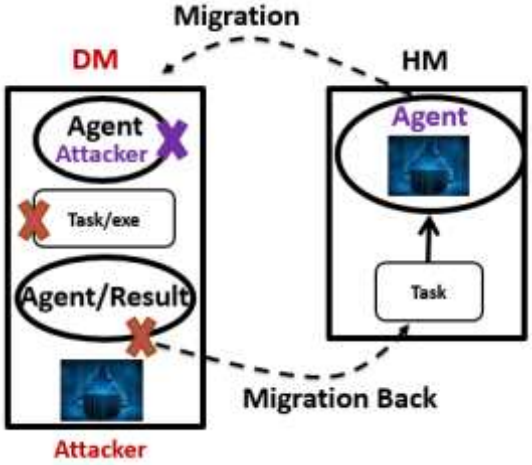

Fig -2: Trade-off between the visiting mobile agent and the visited DM.

Research questions. Three main research questions can be inspired from Figure 2. First, under the assumption that the DM is an attacker and the visiting mobile agent can be

attacked (the carried task itself or the results of the execution of the task), the question of how to enable the mobile agent to carry the protection mechanism is considered important for research. Second, since the visiting mobile agent is executed within the space of the DM, how to isolate this visiting mobile agent from the hosted machine's memory space, so that self-communication can be achieved, is another research question. The final question is how to ensure that the visiting mobile agent came from a trusted HM and will not perform malicious activities at the DM.

Contribution. The use of public key infrastructure is considered an effective way to ensure confidentiality and authentication of the visiting mobile agent at the DM side. However, this approach cannot provide protection to the mobile agent against the DM when it acts as an attacker. By employing a mediator that can be sent (or carried) by the mobile agent and then using this mediator as a function for the encryption key generation process, we can protect the mobile agent, achieving self-protection and self-communication. In general, the contribution of this work is as follows:

- We propose a novel system called the Module-based Security System (MbSS) that ensures the security of the mobile agents in distributed systems. MbSS employs a third key called a mediator in the process of encryption and decryption to achieve self-decryption and self-communication.
- We present a new security metric that depends on the number of security requirements that are achieved. This metric is used for evaluation purpose and involves 13 security requirements in its scale.
- We conduct extensive experiments to evaluate the proposed system compared with four systems proposed previously in the research field of agent security.

The rest of the paper is organized as follows. The related work is presented in section 2. The proposed approach is provided in detail in section 3. The used metrics are presented in section 4, followed by experimental results in section 5. Finally, the paper is concluded in section 6.

2. RELATED WORK

There are two main classes of approaches proposed in the research field on the security of mobile agents: 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.

Many survey papers have explored the approaches on both sides, the most recent of which were in 2019 and in 2017, located in references [3] and [4], respectively. We review some approaches on both sides in this section.

2.1 Protecting visiting mobile agent against malicious DM.

In this category, many techniques are employed to protect the mobile agent against any malicious action that may be applied by the DM.

The co-signing technique relies on hiring an external trusted party to co-sign the migration of the agent. In [5], the preceding DM is considered the external party, which acts as an observer by assuming the responsibility of co-signing the mobile agent. The work in [5] was proposed to give mobile agents resistance against multiple colluding DMs that seek to corrupt the results of execution. Another approach is presented in [6] 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. 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.

The fragmentation-based encryption technique aims at enhancing the performance, where only the sensitive data that may be exploited by a DM are first extracted. These sensitive data are then encrypted. Finally, the encrypted sensitive data are randomized so that only the agent knows the process of returning the correct order. In [7], the bytes of the agent's code are scanned, and 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 [7], the protocol proposed in [8] depends on a fragmentation technique. The difference is that the extraction, encryption, and randomization stages are performed by a TTP.

The Digital Signature (DS) technique is commonly used in secure communication networks and satisfies confidentiality and integrity. It is similar to the code signing technique, but the difference is that it applies a digital signature to the mobile agent itself instead of the carried code. A DS-based approach supported by a checkpoint mechanism is provided in [9]. The objective of the checkpoint mechanism is to guarantee the validity of the mobile agent using fragmentation and defragmentation methods. Based on both DS and the verifying method, another approach is proposed in [10]. In that work, the authors mixed the code signing technique with the DS 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.

Originally, watermarking referred to the process of embedding a watermark within an information entity, such as image [11], audio, video, or text files for copyright protection purposes. In addition, it can be performed on a high-performance computing infrastructure [12] and using

datamining techniques [13]. The authors of [14] exploited 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 attacks them, the embedded watermark is damaged or destroyed. This technique can be enhanced by using agents to save power consumption [15]. However, relying on agent-based software technology has a drawback related to privacy protection required in location-based services [16, 17, 18]. To detect the occurrence of an attack, the watermark is extracted at the HM and compared with the original. The work [14] was developed by the same authors in [19] to be adopted with various kinds of watermarks. 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.

Before you begin to format your paper, first write and save the content as a separate text file. Keep your text and graphic files separate until after the text has been formatted and styled. Do not use hard tabs, and limit use of hard returns to only one return at the end of a paragraph. Do not add any kind of pagination anywhere in the paper. Do not number text heads-the template will do that for you.

Finally, complete content and organizational editing before formatting. Please take note of the following items when proofreading spelling and grammar:

2.2 Protecting visited DM against malicious visiting mobile agent

Similar to the previous category, many techniques are used to protect the visited DM against malicious mobile agents.

Code signing is a technique that attempts to ensure the integrity of the code that is executed on the DM platform. It tunes with both the one-way hash functions and the DS concepts to ensure that no modification is made to the code. Therefore, this technique assumes that the creator of the code is trusted. The authors of [20] provide proof of the resistance of this technique, which is used in ActiveX controls and Java applets. An enhanced verification-based approach was introduced by Malik et al. [21]. Their key idea depends on using whitelists and blacklists 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 granted full permissions to be executed. Otherwise, it will be frozen.

In the Proof-Carrying Code (PCC) technique, the creator of the code marks the code (i.e., generates a proof attached to the original code), so that any modification that occurs will be detected and the code will not be allowed to execute. Compared to the code signing technique, the PCC is better in terms of time and computation costs. This is because the PCC does not require cryptography for the digital signature. In [22], the authors proposed a foundational PCC, in which the code was 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 [23], where the major concern is allowing dynamic access to the platform of DM, with a tolerance of strict proof representation.

3. PROPOSED SYSTEM

To ensure agent security, we must ensure that (1) the visiting mobile agent is benign (i.e., not malicious) and (2) the DM is prevented from performing malicious actions on the visiting mobile agent (i.e., the DM is honest). For this purpose, we must ensure the following security requirements: authentication, authorization, confidentiality, availability, and integrity. To achieve this, we support middleware (or an agent manager, such as Concordia) by some modules that can implement these five security requirements. Figure 3 shows the deployment architecture of the proposed system.

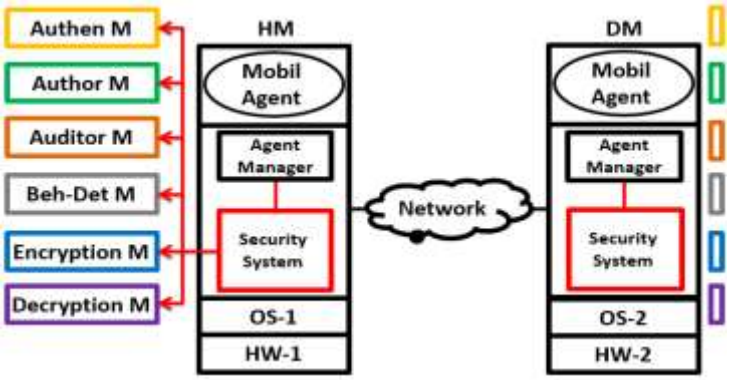

Fig -3: Deployment architecture of the proposed system.

3.1 System architecture

The proposed system consists of six modules, as shown in Figure 4.

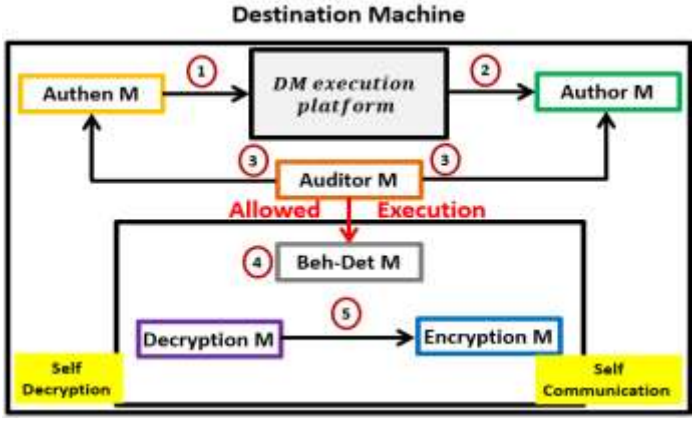

Fig -4: Architecture of the proposed system.

In terms of the communicated modules, MbSS achieves self-decryption (or self-protection) and self-communication. Each module has its own objective. They are as follows:

1. Authen M, used to satisfy the authentication security requirement, where the visiting mobile agent must come from a trusted machine.
2. Author M, used to satisfy the authorization security requirement, where the visiting mobile agent must not exceed its privileges.
3. Auditor M, used to satisfy the accountability and non-repudiation security requirements (i.e., traceability of malicious actions to attackers, holding them responsible for their actions and ensuring that the agent platform which sends agent to the DM cannot deny that it is the owner of the mobile agent).
4. Behaviour detector M, used to detect the behaviour of the visiting mobile agent, so that if any malicious behaviour occurs, the agent is terminated.
5. Encryption M and Decryption M, used to satisfy the confidentiality security requirement, where the information collected by the mobile agent must be kept secret.

3.2 Achieving security requirements practically

To achieve the security requirements, many steps are performed at both the HM and the DM. Before presenting the details, the following definitions are presented.

Definition 1. S_{task}^{PR} denotes the set of the privileges that the agent must have to perform a task at the DM. It is defined as

$$S_{task}^{PR} = \begin{cases} \langle PR_1 \rangle \\ \langle PR_2 \rangle \\ \langle PR_3 \rangle \\ \dots \\ \langle PR_n \rangle \end{cases} | n \text{ is number of privileges} \quad (1)$$

Definition 2. key_{ASE} refers the session key of the AES symmetric algorithm. It is generated based on another key called the mediator key $(\text{key}_{\text{mediator}})$, so that

$$\text{key}_{\text{ASE}} = \text{function}\ (\text{key}_{\text{mediator}}) \quad (2)$$

Definition 3. $\text{key}_{\text{public}}^{\text{DM}}$ denotes the public key of the DM. Any message encrypted using $\text{key}_{\text{public}}^{\text{DM}}$ is decrypted using the private key of the DM, denoted as $\text{key}_{\text{private}}^{\text{DM}}$.

- Based on the previous three definitions, the steps performed at the HM are as follows:
  1. The agent owner creates a mobile agent.
  2. The task to be performed at the DM and its corresponding set of privileges are defined.
  3. $\text{key}_{\text{mediator}}$ is defined.
  4. The function of generating key_{ASE} is defined.
  5. Both S_{task}^{PR} and $\text{key}_{\text{mediator}}$ are encrypted using $\text{key}_{\text{public}}^{\text{DM}}$ as follows:

$$\overline{\text{key}_{\text{mediator}}} = \text{enc} \begin{cases} \text{key}_{\text{public}}^{\text{DM}} \\ \text{key}_{\text{mediator}} \end{cases} \quad (3)$$

$$\overline{S_{task}^{PR}} = \text{enc} \begin{cases} \text{key}_{\text{public}}^{\text{DM}} \\ S_{task}^{PR} \end{cases} \quad (4)$$

  6. The hash of the task to be executed at the DM is calculated, and both (i.e., the task and its hash) are encrypted using key_{ASE} as follows:

$$\overline{task} = \text{enc} \begin{cases} \text{key}_{\text{AES}} \\ task \end{cases} \quad (5)$$

$$\overline{hash} = \text{enc} \begin{cases} \text{key}_{\text{AES}} \\ hash(task) \end{cases} \quad (6)$$

  7. The mobile agent starts the migration to the DM, carrying

$$Mobile\ Agent = \begin{cases} \overline{task} = \text{enc} \begin{cases} \text{key}_{\text{AES}} \\ task \end{cases} \\ \overline{hash} = \text{enc} \begin{cases} \text{key}_{\text{AES}} \\ hash(task) \end{cases} \\ \overline{S_{task}^{PR}} = \text{enc} \begin{cases} \text{key}_{\text{public}}^{\text{DM}} \\ S_{task}^{PR} \end{cases} \\ \overline{\text{key}_{\text{mediator}}} = \text{enc} \begin{cases} \text{key}_{\text{public}}^{\text{DM}} \\ \text{key}_{\text{mediator}} \end{cases} \end{cases} \quad (7)$$

In other words, if any party (including the DM) wants to obtain key_{ASE}, it must know the function that takes $\text{key}_{\text{mediator}}$ as an input to generate key_{ASE}. Here, the

decryption module knows this function. Figure 5 shows the start of mobile agent migration.

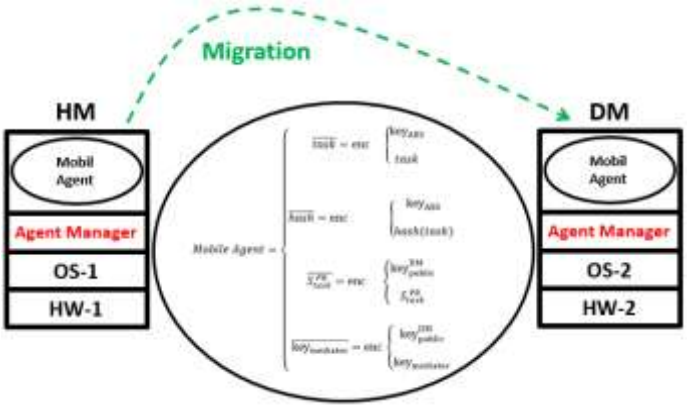

Fig -5: Starting migration of mobile agent.

- The steps performed at the HM are as follows:
  1. The decryption module sends a request, asking the execution platform (i.e., DM) to decrypt $\overline{\text{key}_{\text{mediator}}}$ .The execution platform processes the request and decrypts it according to the following formula:

$$\text{key}_{\text{mediator}} = \text{dec}\begin{cases}\overline{\text{key}_{\text{mediator}}} \\ \text{key}^{\text{DM}}_{\text{private}}\end{cases} \quad (8)$$

This ensures the authentication security requirement (i.e., the DM ensures that the visiting mobile agent comes from a trusted source).

  2. The decryption module sends a request, asking the execution platform to decrypt $\overline{S^{PR}_{task}}$. The execution platform processes the request and decrypts it according to the following formula:

$$S^{PR}_{task} = \text{dec}\begin{cases}\overline{S^{PR}_{task}} \\ \text{key}^{\text{DM}}_{\text{private}}\end{cases} \quad (9)$$

This ensures the authorization security requirement (i.e., the DM knows exactly what privileges the visiting mobile agent has). This gives the DM the ability to detect any malicious activity that the visiting mobile agent may perform outside its privileges.

  3. After receiving $\text{key}_{\text{mediator}}$, the decryption module decrypts it to obtain key_{AES} based on the function by which key_{AES} is generated. The decryption module knows this function.

After obtaining key_{ASE}, the following steps can be performed.

  4. The decryption module included in the agent manager decrypts both $\overline{task}$ and $\overline{hash}$ according to the following formula:

$$task, hash = \text{dec}\begin{cases}\overline{task} \\ \overline{hash} \\ \text{key}_{\text{ASE}}\end{cases} \quad (10)$$

  5. The hash of the decrypted task is calculated and verified according to the value of the decrypted hash by the following formula:

$$hash(dec[task]) \stackrel{?}{=} dec(hash) \quad (11)$$

If they are matched, then three security requirements (verification, integrity, and confidentiality) are ensured.

  6. After the correct matching, the task is executed at the DM. Both the results of the executed task $(results)$ and $\text{key}_{\text{mediator}}$ are encrypted using the public key of the HM ($\text{key}^{\text{HM}}_{\text{puplic}}$). This is accomplished by the communication between the decryption and encryption modules. The following formula shows the encryption process to protect the results of the executed task (from malicious altering) and the mediator key (from revealing to the DM).

It is worth mentioning that $\text{key}_{\text{mediator}}$ can be decrypted by the public key of the next DM included in the itinerary of the mobile agent if it visits more than one DM. In this case, the first DM is considered the HM, and the next DM is the current DM. All previous steps are performed where

$$DM_1 = HM \text{ and } DM_2 = DM_{current} \quad (13)$$

Figure 6 illustrates the steps performed at the DM to achieve self-decryption and self-communication.

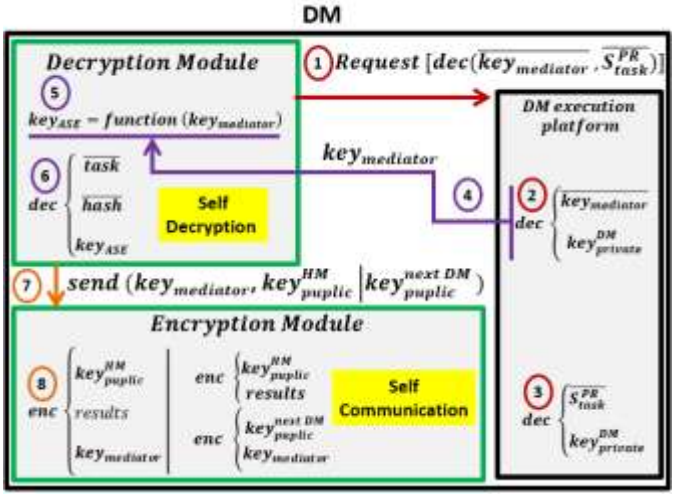

Fig -6: Achieving self-decryption and self-communication at the DM.

4. USED METRICS

Two kinds of metrics are used for evaluation in this work: security metrics and performance metrics.

4.1 Security metrics

We depend on the security requirements that may be achieved to measure the security level. The system under evaluation gains one point if a security requirement is guaranteed. Since the DM tries to ensure its security against the visiting mobile agent (considering this agent as an attacker) and the visiting mobile agent tries to protect itself against the DM (considering it as an attacker), we consider the security requirements that should be achieved on both sides (i.e., DM and mobile agent). Table 1 summarizes the security requirements involved in the measurement.

Let SL denote the security level; the maximum value is 13 according to the used security requirements, and the minimum value is 0. Then,

$$SL(value) \in [0, 13] \qquad (13)$$

The higher the SL value, the higher the security level achieved by a given security system, and vice versa.

Table -1: Used security requirements.

Abbr./Name	Description
Traditional security requirements	
C (confidentiality)	Ensures that information carried by the mobile agent must be kept secret and only authorized parties can access it.
I (Integrity)	Guards 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)	Ensures that actions performed on a DM are traceable to the agent that 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 of the agent that 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 it is the owner of the specific information and agent.
Ve (Verification)	Ensures that only the authenticated mobile agent is permitted access into the DM and verifies the code of the migrated agent from the HM before execution.
Security requirements for isolation from the DM's space	
SD (Self-Decryption)	Ensures that the agent manager is supported by a special module to perform any decryption task.
SC (Self-Communication)	Ensures that the agent manager is supported by special modules and the modules communicate with each other directly without a third party (i.e., DM's platform).

4.1 Performance metrics

Time is the most important performance metric. In this context, we define all processes that require a specified time to end: encryption, decryption, hash calculation, and function calculation. Each process has its corresponding time. The total time that results from the sum of all times represents the performance metric, defined as

$$T_{total} = T_{enc} + T_{dec} + T_{hash} + T_{function} \qquad (14)$$

The lower the T(total) value, the higher the performance level, and vice versa.

4. EXPERIMENTAL RESULTS AND EVALUATIONS

We evaluate the proposed system based on the metrics defined previously and compared with four works. The four works are shown in Table 2.

Table -2: Works involved in comparison.

Work No	Work ref	Used technique	Purpose
1	[6]	Co-signing (CoG)	Protecting the mobile agent
2	[10]	Digital Signature (DS)	
3	[21]	Code signing (CodG)	Protecting the DM
4	[23]	PCC	

SL-based evaluation. In the context of this evaluation, we count the points that each work gains according to the security requirements achieved. Table 3 shows the calculated points.

Table -2: Gained points.

Work / Security req	CoG	DS	CodG	PCC	MbSS
C	1	1	1	0	1
I	0	1	0	1	1
A	0	0	0	0	1
An	0	0	0	0	1
Ac	0	0	0	0	1
Au	1	1	1	0	1
Ar	0	0	0	0	1
At	1	1	0	1	1
As	0	0	0	0	1
Non-R	0	1	0	0	1
Ve	0	1	1	0	1
SD	0	0	0	0	1
SC	0	0	0	0	1
Total Sum	3	6	3	2	13

Figure 7 illustrates the results obtained in Table 3.

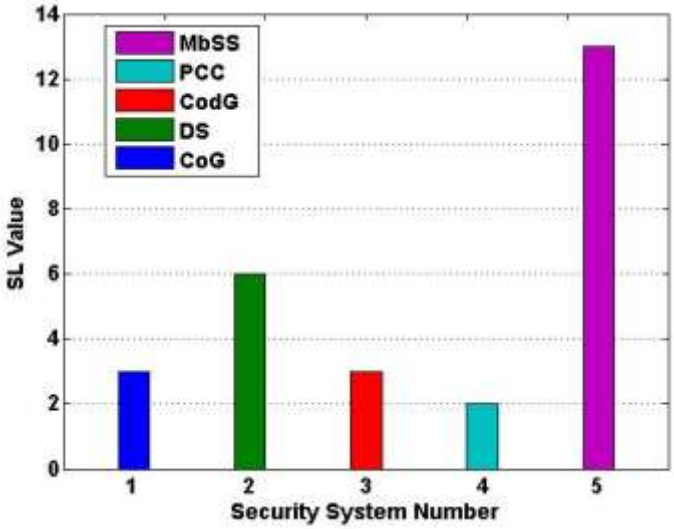

Fig -7: SL-based comparison.

Discussion. The proposed security system, MbSS, achieves the highest security level. There are several reasons behind this. The use of encryption based on key^{DM}_{public} and decryption based on $key^{DM}_{private}$ ensures traditional security requirements and authentication. In addition, defining the privileges of the visiting agent ensures authorization. Since the auditor module monitors the behaviour of the mobile agent within the space of the DM, the rest of the "six A's" security requirements are guaranteed. This is because the agent is traceable and cannot deny any of its actions. Finally, MbSS was originally designed to achieve self-decryption and self-communication. In terms of ranking, DS followed the proposed system, achieving 6 points. Since the DS technique relies on hashing and then encrypting it using the public key (of the receiver) and the private key (of the sender), the C, I, and Au security requirements are achieved. The enhanced technique used ensures three additional security requirements (i.e., At, Non-R, and Ve). Both CodG and CoG have the same security level, where each hit a score of 3. PCC has the poorest security level with only 2 points. This is because it targets performance rather than security, where no encryption or decryption stages are involved at all.

T_{total}-based evaluation. In the context of this evaluation, we involve only CoG and DS in the comparison. This is because they are considered the most common methods used for protecting mobile agents. We compare the three systems according to the total time spent increasing the size of the sent agent, as shown in Figure 8.

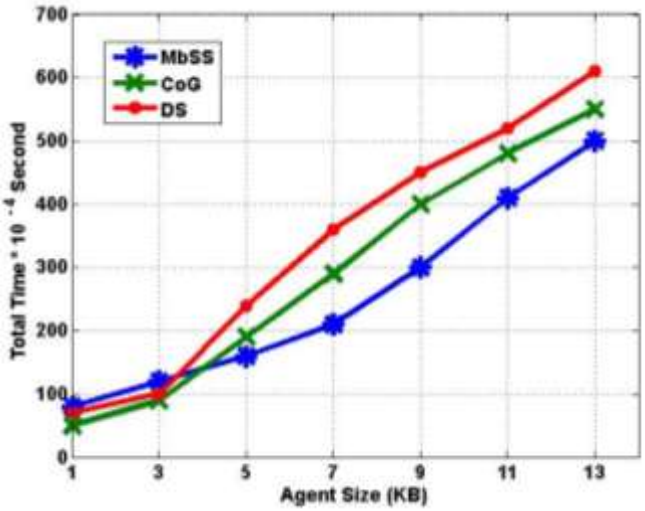

Fig -8: T_{total} -based comparison.

Discussion. As shown in Figure 8, the proposed MbSS performs worst, followed by DS, and CoG performs the best at the beginning (i.e., when the size of the agent is small). This is because CoG does not require the same computation as DS and MbSS, which results in a short execution time. Since the time required to perform a digital signature on the whole code of the mobile agent is less that the time needed to perform the different processes (i.e., encryption, decryption, hashing, and function of the mediator key), DS outperforms MbSS. However, when the size of the mobile agent increases, the whole scenario changes. DS performs worst because the time needed to perform the hashing, encryption, decryption, and matching processes is long because of the large size of the mobile agent. CoG comes in second because of the time needed to communicate to the third part responsible for the entire process of signing and sending back the signed code. The system proposed in this work performs best. This is because the digital signature (including the hashing, encryption using the public key of the DM, and decryption using the private key of the DM) is performed only on the task carried by the agent, not on the whole code of the mobile agent. Second, the key of the encryption session is generated locally using the function associated with the mediator key, not by the encryption and decryption process. This means that the time needed to obtain the session key is shorter than that needed to obtain the same key in DS and CoS. Finally, not all the code of the agent is encrypted in MbSS (i.e., the itinerary and other parameters), which means that a kind of fragmentation is performed, which leads to less time required to end the whole mission.

5. CONCLUSION

Security is considered the most important issue in agent-based software technology. The security problem arises from the fact that when a mobile agent caries a task to perform on a remote destination machine, the execution of the mobile agent is performed within the space of the hosted machine. This means that if the destination machine is an attacker, the whole task is compromised, or the agent will return with false results. In addition, the destination machine has the right to ensure that the visiting mobile agent will not bring damage to its platform. In responding to this problem, we present the Module-based Security System (MbSS). The system consists of six modules: authentication, authorization, auditing, behaviourbehaviour detection, encryption, and decryption. The communication among the modules is conducted within the manager of the agent (middleware installed on the destination machine). In addition, the decryption process is performed using a mediator key, which ensures that the process of generating the encryption key of the session is not controlled by the destination machine. Based on 13 security requirements—confidentiality, integrity, availability, anonymity, accountability, authorization, accounting, non-repudiation, assurance, verification, self-decryption, and self-communication—MbSS outperforms similar systems, achieving a score of 13. In terms of performance time, MbSS also performs better than similar systems when the size of the agent increases.

REFERENCES

[1] 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.

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

[3] Bandar Alluhaybi, Mohamad Shady Alrahhal, Ahmed Alzhrani and Vijey Thayananthan, “A Survey: Agent-based Software Technology Under the Eyes of Cyber Security, Security Controls, Attacks and Challenges” International Journal of Advanced Computer Science and Applications(IJACSA), 10(8), 2019.

[4] Bagga, Pallavi, and Rahul Hans. "Mobile agents system security: a systematic survey." ACM Computing Surveys (CSUR) 50.5 (2017): 1-45.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

[11] Al-Rahal, M. Shady, Adnan Abi Sen, and Abdullah Ahmad Basuhil. "High level security based steganoraphy in image and audio files." Journal of theoretical and applied information technology 87.1 (2016): 29.

[12] Fouz, Fadi, et al. "OPTIMIZING COMMUNICATION AND COOLING COSTS IN HPC DATA CENTER." Journal of Theoretical and Applied Information Technology 85.2 (2016): 112.

[13] Alrahhal, Mohamad Shady, and Adnan Abi Sen. "Data mining, big data, and artificial intelligence: An overview, challenges, and research questions." (2018).

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

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

[16] 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.

[17] 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).

[18] Alrahhal, Mohamad Shady, et al. "AES-route server model for location based services in road networks." Int. J. Adv. Comput. Sci. Appl 8.8 (2017): 361-368.

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

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

[21] 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.

[22] 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.

[23] 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.

AUTHORS

Bandar Alluhaybi: received the B.S. degree in computer science from King Abdul Aziz University in 2009, and the M.S. degrees in Engineering System Management from St Mary's University, San Antonio, TX, USA in 2012. He currently stydies the Ph.D. degree in computer science at King Abdoulaziz University, KSA. He is currently a Lecturer with the Computer Science Department, Prince Mugrin University. His current resarch interests include information security, computer networs, networks security, , big data, and high performance computing.

Mohamad Shady Alrahhal: received PHD from king abdulaziz university, KSA, department of computer science, college of computing and information technology (2018). Received master degree from Damascus University, Syria, (2013). Received a degree of computer engineering from Albath University, Homs, Syria, (2011). Currently, he interests in agent based software technology, artificial intelligent, cyber security, imageand video processing, and risk management. Practically, he was the director of the IT department of IoT company, and risk management manager.

Ahmed Alzahrani: received the B.S. degree in computer science from King Abdul Aziz University in 2000, and the M.S. degrees in information security from University of Glamorgan, Cardiff, UK in 2005. He received the Ph.D. degree in computer networks from University of Bradford, UK in 2009. He is currently an Associate Professor with the Computer Science Department, and the current Vice Dean of Deanship of Graduate Studies for academic affaris, King Abdul Aziz University. His current resarch interests include computer networs, networks security, quality of service routing, quantum computing, deep learning, big data, in-memory computing, and high performance computing.

Vijey Thayananthan: is a Professor at Computer Science Department in the King Abdulaziz University, Jeddah, KSA. He obtained his Ph.D. in Engineering (Digital communication engineering) from Department of Communication Systems, University of Lancaster, UK, in 1998. Since 2000, he had been working as a Research engineer and senior algorithm development engineer in Advantech Ltd, Southampton University Science Park, UK and Amfax Ltd, UK respectively. His research interests include wireless communication algorithm design and mobile communication analysis, security management of communication network and big data, computer security and wireless sensor network. He has been a full-time member of the Institution of Engineering Technology (IET), UK since 2005.