IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 17, 2020, accepted July 8, 2020, date of publication July 14, 2020, date of current version July 24, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3009245

Dummy-Based Approach for Protecting Mobile Agents Against Malicious Destination Machines

BANDAR ALLUHAYBI, MOHAMAD SHADY ALRAHHAL, AHMED ALZAHRANI, (Member, IEEE), AND VIJEY THAYANANTHAN

Department of Computer Science, Faculty of Computing and Information Technology, King Abdulaziz University (KAU), Jeddah 21441, Saudi Arabia

Corresponding author: Bandar Alluhaybi (balluhaybi0002@stu.kau.edu.sa)

ABSTRACT Recently, agent-based software technology (ABST) has received widespread attention from the research community and users. However, security issues facing ABST are critical. When a mobile agent migrates from their home machine to perform tasks, the agent becomes vulnerable to attacks by the destination machine, which has full control over the visiting mobile agent. To address this security issue, we propose a dummy task selection (DTS) approach to protect the mobile agent by confusing the attacker (destination machine) with regard to distinguishing the real task among dummy attacks. Considering that side information may be employed by the attacker to perform advanced attacks, we introduce improved DTS as an enhancement of the DTS approach. The improved DTS approach generates strong dummy tasks based on execution probabilities that lead to the highest entropy. Unlike previous approaches, the improved-DTS approach performs the full protection mechanism at the home machine, which in turn limits the ability of the attacker to control the visiting mobile agent. Compared to previous approaches, both the DTS and improved-DTS methods achieved better performance and higher resistance to advanced active attacks such as alternation, collusion, and DoS attacks.

INDEX TERMS Agent, attack, destination machine, dummy, home machine, task.

I. INTRODUCTION

A. IMPORTANCE OF SOFTWARE AGENTS

One of the most important software technologies 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 on behalf of a network user [1], [2]. Compared to other technologies such as message passing (MP) [3], remote procedure call (RPC) [4], and code on demand (CoD) [5], ABST achieves better performance in distributed systems in terms of scalability (for both increasing numbers of users and sizes of data), manipulation (at both access latency and tuning time levels), and network latency [2], [6]. Moreover, ABST is highlighted in regard to solving transmission challenges when addressing big data, especially if the data are of images or videos [7], [8]. In other words, instead of sending the big data to the processing machine, an agent can be sent to the place where the big data are located, therein processing the data locally and then returning with the results only, which in turn contributes to reducing the network overhead, as illustrated in Figure 1.

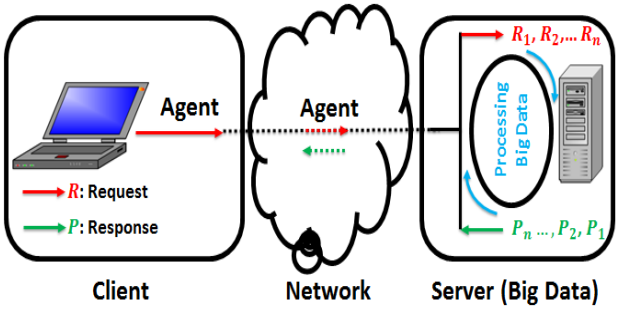


FIGURE 1. Solving transmission challenge by ABST.

B. STATEMENT OF PROBLEM

Both the strengths and weaknesses of ABST come from the features of the agent as follows: (1) Adaptability, which

The associate editor coordinating the review of this manuscript and approving it for publication was Junaid Arshad.

means platform independence, whereby the agent can be executed on different machines with different operating systems; (2) Transparency and accountability, which means that the software agent runs on behalf of its owner, and the owner can ask the agent about both its current location and what has been accomplished; (3) Ruggedness, which enables the agent to run given low or high resources, interpreting different data formats; and (4) Mobility, which is a unique property of this technology. This means that the agent can move from one machine, called the home machine (HM), to another machine, called the destination machine (DH), performing a specific task there and then returning to the original machine with the results [9], [10]. In other words, the agent is goal driven. Figure 2 illustrates the classical scenario of performing a mission by an agent.

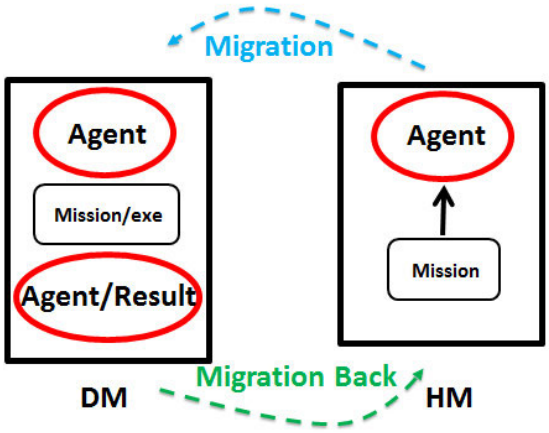


FIGURE 2. Classical scenario of performing a mission by an agent.

The first three features of ABST (i.e., adaptability, transparency and accountability, and ruggedness) make it a powerful technology. However, the mobility feature makes it vulnerable to attacks from a security point of view. This is because the migrated agent can be attacked by a man-in-the-middle attack or even by another malicious agent. The security problem is accentuated in regard to talking about the DM as an attacker. Since the DM has full control over the visiting mobile agent, it has the ability to tamper with the results of the executed mission, which in turn leads to loss of result integrity, or even the whole code (behavior of the agent). Moreover, the attacker (DM) can apply advanced attacks, such as alternation attacks [11], collusion attacks [12], [13], and denial of service (DoS) attacks [2], [14]. Furthermore, some malwares and advanced persistent threats can be easily distributed to steal data from the visiting agents [15], [16]. This in turn makes solving the security issues facing ABST pressing.

C. MOTIVATION

The importance of presenting an effective solution to ensure the integrity of the results is highlighted when an agent is used to perform sensitive tasks. For example, in smart cities, where ensuring safety is very important, smart warning systems (SWSs) are used to collect data about the bridges, cranes, and swings in kids' play areas, which are located in different locations of the smart city [17], [18]. When using agents to migrate to DMs (computers connected to cameras) and a video magnification task [19]–[21] is assigned to the agents to check if the vibration of the items is normal or not, returning unmodified results is critical to the decision-making center taking proper corresponding steps. If data are compromised, this can lead to disasters, and many people may lose their lives. Figure 3 illustrates the key idea behind this example.

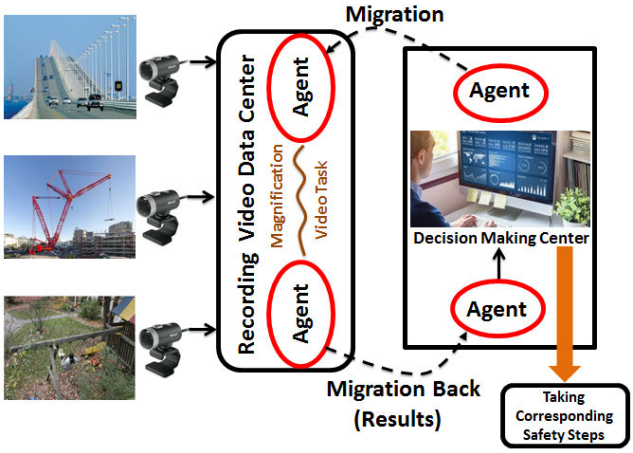


FIGURE 3. Agent-based SWS in smart cities.

To protect mobile agents and to ensure the integrity of their task results against malicious DMs, researchers have proposed many approaches such as result partial encapsulation (RPE) [22], obfuscated code (OC) [23], environmental key generation (EKG) [24], co-signing [25], and fragmentation-based encryption (FBE) [26]. However, the main drawbacks of these techniques are related to their low performance and high complexity since all of them (except for the EKG and OC techniques) rely on encryption and decryption concepts. Moreover, the malicious DM can keep a copy of the secret encryption key, which in turn leads to a large security gap in such approaches since the decryption process is performed within the space of the hosted machine (i.e., the DM, which is the attacker itself) [27]. The problem with the EKG-based technique is related to generalization, where it limits the execution of the visiting mobile agent to the conditions that must be satisfied on the DM side. In other words, it is unsuitable for certain applications since it may lead to the blocking of the execution of the visiting agent [28]. Furthermore, all the approaches mentioned above suffer from low resistance to advanced attacks, such as alternation attacks, collusion attacks, and denial of service (DoS) attacks, according to previous works [2], [28]. Beyond that, from a security point of view, the whole protection mechanism should be performed on the HM side (user or agent owner side). However, all previous protection approaches have complementary parts

that are executed at the DM such as the decryption process in the FBR approach.

D. CONTRIBUTION

Since the visiting mobile agent is controlled by the DM and that the task is executed within its space, one way to ensure the integrity of the task's results is to confuse the DM (as an attacker) and prevent it from determining the executed task. This can be achieved by executing dummy tasks so that the attacker cannot recognize the real task among the dummies. In this paper, we introduce a novel dummy-based approach to protect mobile agents, which combines two main quality attributes (high performance and high resistance to attacks) and can be adopted with all agent-enabled applications. The main contributions of this paper are as follows:

- To protect the behavior of the mobile agent as well as ensure the integrity of the results, we propose a novel protection approach for mobile agents called dummy task selection (DTS). This technique relies on generating dummy tasks to be executed on the attacker side (destination machine) so that the real task cannot be recognized among the dummy tasks.
- Considering resistance against advanced attacks, we improve the DTS approach. The improved DTS approach generates strong dummy tasks based on both the execution probability and the type of real task.
- We introduce a novel security metric to estimate the protection level that is achieved. The security metric mainly depends on the entropy to quantify the amount of protection, therein considering the amount of violation caused by the attacker side.

The remainder of the paper is structured as follows. The related work is reviewed in section II. The system architecture along with the proposed approaches are presented in section III. Section IV discusses the security analyses. In section V, the metrics used are introduced, followed by the experimental results and evaluations in section VI. Finally, the paper is concluded in section VII.

II. RELATED WORK

In this section, we provide an overview of mobile agents and their life cycle. Then, we review some security approaches that were proposed to protect mobile agents.

A. OVERVEW OF MOBILE AGENT

As we mentioned earlier in the introduction section, the agent has the ability to roam among different machines connected to the internet called DMs. Therefore, the HM does not restrict where it is written, as illustrated in Figure 4.

As shown in Figure 4, there is one HM and three different DMs, and each machine has its own operating system (OS) and hardware (HW) specifications. The mobile agent is created (or written) by the owner at the HM using an agent manager, which is uniform and must be installed on all machines. Several agent managers, such as Concordia, Java Agent Development Framework (JADE) and Agelets,

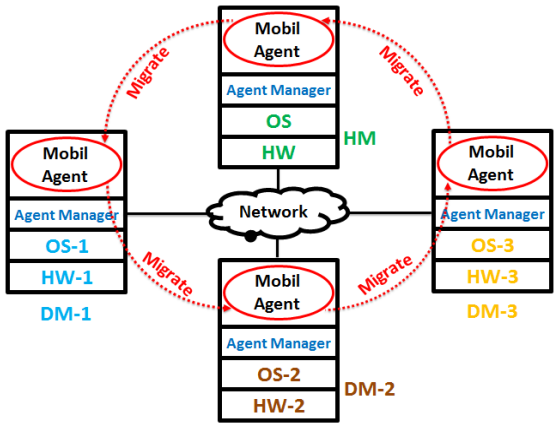


FIGURE 4. Migration of mobile agent.

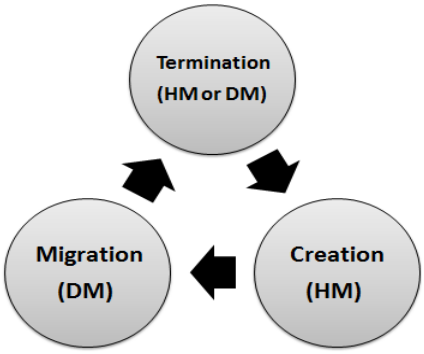


FIGURE 5. Life cycle of the mobile agent.

are available. Using an agent manager and after agent creation, an itinerary that includes DMs is defined, as well as the mission that will be executed there. According to this, three main stages are involved in the lifecycle of the mobile agent, creation, migration, and termination, as shown in Figure 5.

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).

B. PROTECTING MOBILE AGENT

In the security field of ABST, researchers have classified security approaches into two main categories according to the surveys conducted in the works [2], [28]: (1) protecting agent platform (PAP) and (2) protecting mobile agent (PMA) approaches. Table 1 shows the main difference between the two categories.

TABLE 1. Security approaches classification.

Category	Attacker	Victim
PAP	Malicious visiting mobile agent	Destination machines
PMA	Destination machines	Visiting mobile agent

Since we are concerned in this work about the PMA category, we review some related approaches. The key idea

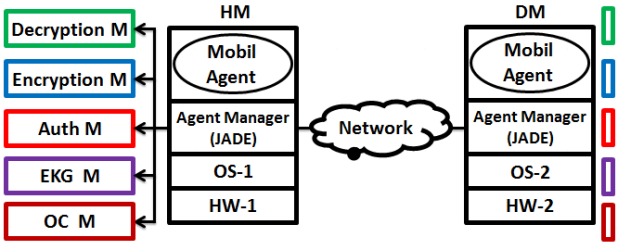


FIGURE 6. General architecture of some PMA approaches.

of each approach reviewed in this section is to equip the agent manager with a specific module that is responsible for applying the intended protection approach, as shown in Figure 6.

1) RESULT PARTIAL ENCAPSULATION (RPE)

This technique is designed to detect any changes that might occur regarding the results of an executed task 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 to the agent's code to provide confidentiality using encryption based on a secret key. Relying on the RPE technique, Yee at el. proposed the approach in [22]. 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 a message authentication code (MAC). Then, encapsulating the MAC with a message that represents the results of the task execution generates a partial result authentication code (PRAC). Therefore, encryption, decryption, and authentication modules are used to support the agent manager (JADE in Figure 6 above).

2) COSIGNING

This technique relies on hiring an external trusted party to co-sign the migration of the agent. The key idea of the work proposed by Linna and Jun [25] is that after producing the results, the DMs encapsulate them with information about the task performed by the mobile agent. Then, the entire encapsulated package is encrypted and sent to the next DM. When the mobile agent reaches the next DM, a comparison is performed between the generated results and the task information to discover any attack that may have occurred. Therefore, encryption and decryption modules are used to support the agent manager.

3) FRAGMENTATION BASED ENCRYPTION (FBE)

This technique proposed by Srivastava and Nandi [26] aims to enhance 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. 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 as well as the results when returning to the HM. Therefore, similar to the co-signing approach, encryption and decryption modules are essential.

4) OBFUSCATED CODE (OC)

In this technique, the mobile agent travels through a series of DMs of 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 numbers), the behavior of the mobile code is protected. The key idea is to perform 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, Badger *et al.* [23] proposed a 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. Therefore, the agent manager is supported by an OC module.

5) ENVIRONMENTAL KEY GENERATION (EKG)

This technique relies on the principle that "the execution is not allowed unless some environmental conditions are satisfied at the DM". In [24], 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. Therefore, an EKG module is needed to support the agent manager.

III. THE PROPOSED SYSTEM ARCHETECTURE

This section is organized so that the threat model is presented first. Then, the main proposed approach is introduced. Finally, an enhancement of the main approach is provided.

The general architecture of the proposed system is illustrated in Figure 7.

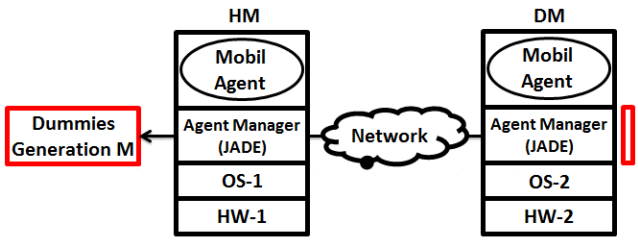


FIGURE 7. General architecture of the proposed system.

As shown in Figure 7, the agent manager is supported by a dummy generation module. This module is responsible for generating dummy tasks to be executed at the DM. The final goal of this module is to protect the behavior of the agent (i.e., the code of the real executed task, referred to as the integrity of execution) as well as ensure the integrity of the results.

A. THREAT MODEL

The attacker is the DM at which the task of the visiting mobile agent is executed, as shown in Figure 8 below.

We consider the side information possessed by the attacker. This term refers to the fact that the attacker knows the behavior of the agent (i.e., the loops, if-then-else and other statements included in the code of the executed mission) as well as the frequency of the executed task. Based on this side information, the objective of the attacker is to tamper with the code of the executed task or to modify the results of the execution. To end this, we assume that the attacker has capabilities to perform different attacks, which are summarized in Table 2.

TABLE 2. Cababilities of the attacker.

Capability NO.	Attack name	Attack type
1	Alternation attack	
2	Collusion attack	Active
3	DoS attack	

In an alternation attack, the malicious DM targets either the execution integrity or the result integrity of the task. Figure 8 illustrates the main concept of the alternation attack.

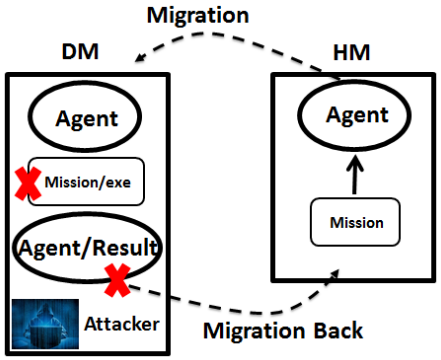


FIGURE 8. The attacker.

An alternation attack can be converted into a more dangerous attack, called a collusion attack [2], [28]. In collusion attacks, two malicious DMs (or more) collude to modify the results. Formally, let (n) denote a number of DMs ($DM_1, DM_2, \ldots, DM_{(n-1)}, DM_n$), which form a series of malicious DMs. Let ($Res_i, \tilde{Res}_{Ai}$) denote the original result of the executed mission and the attacked mission (or modified) by the i^{th} DM, respectively. Then,

$$\tilde{Res}_{A1} = \tilde{Res}_{A2} = \cdots = \tilde{Res}_{Ai} \mid 0 < i \leq n \qquad (1)$$

In other words, the HM is tricked when receiving the same modified results generated at the malicious DMs. Therefore, each DM involved in the malicious series is seen as honest in the eyes of the HM. Figure 9 illustrates the concept of the collusion attack.

In a DoS attack, the mobile agent performs a task referred to as a time-sensitive task *TST*. The *TST* is attached to a deadline (T_{dl}), and both of them are carried by the mobile agent to be executed at the DM within the period of the T_{dl}. The result of the *TST* becomes invalid if the execution time of the *TST* exceeds the predefined T_{dl} even if it is correct (i.e., it was not modified) [2], [28]. Therefore, the goal of the malicious DM is to lengthen the execution time or modify T_{dl} to ($\tilde{T}_{dl}$), as shown in Figure 10.

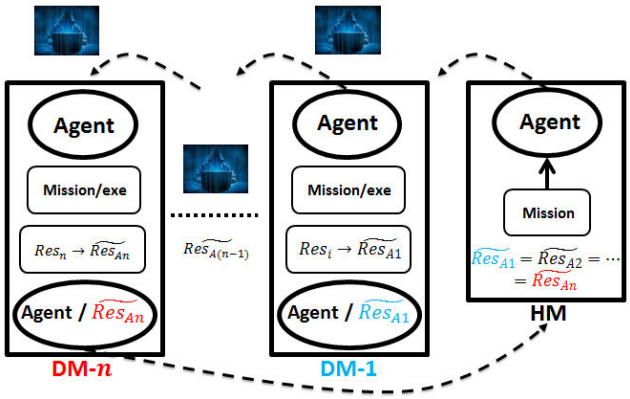


FIGURE 9. Concept of the collusion attack.

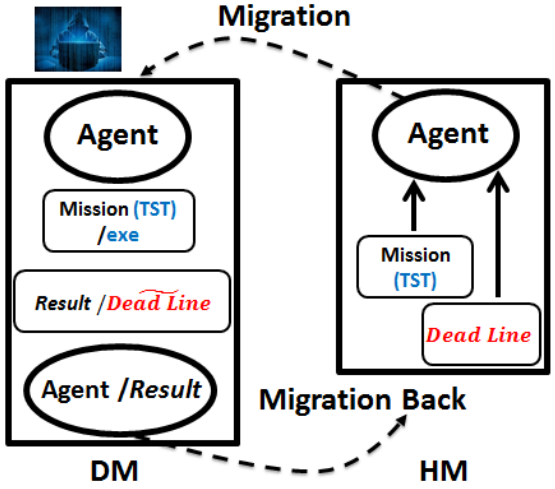


FIGURE 10. Concept of DoS attack.

B. ROLE OF DUMMY GENERATION MODULE

The final goal of our proposed module is to generate strong dummy tasks to be executed at the DM, which in turn aims at guaranteeing both the execution integrity and the result integrity of the visiting mobile agent. Executing dummy tasks along with the real tasks limits the ability of the attacker to recognize the real code (or the actual generated result) among the dummies for malicious modification purposes. To this end, the dummy generation module applies an enhanced dummy task selection approach (improved DTS) to select strong dummies. We first introduce the DTS approach and then the improved-DTS approach, as described below.

1) DUMMY TASK SELECTION (DTS) APPROACH

Definition 1: $DB_{tasks}^{historical}$ refers to a database at the HM. The database includes historical tasks that have been executed at different DMs. Formally, it is given by the following formula:

$$DB_{tasks}^{historical} = \bigcup_{i=m}^{i=1} Task_m, \quad m = size\,(DB) \tag{2}$$

Definition 2. $Task_{freq,dl}^{ref,DM^i}$ refers to a task stored in the previous DB. Each task is modeled so that it includes historical information. This historical information mainly includes the frequency of the execution of the task (*freq*) and at which i^{th} DM is executed. (*ref*) is used to distinguish the task, and (*dl*) is used to mark the task as a normal task or time-sensitive task as follows:

$$Task_{freq,\,dl}^{ref\,,\,DM^i} = \begin{cases} NT, & if\ dl = \infty \\ TST, & if\ dl = value \end{cases} \quad i = 1, 2, \ldots, or\ n \tag{3}$$

Here, each DM has a number of tasks that are performed within its space previously, and each task has its own frequency number. All of this information is stored in a DB at the HM side. Figure 11 illustrates the idea.

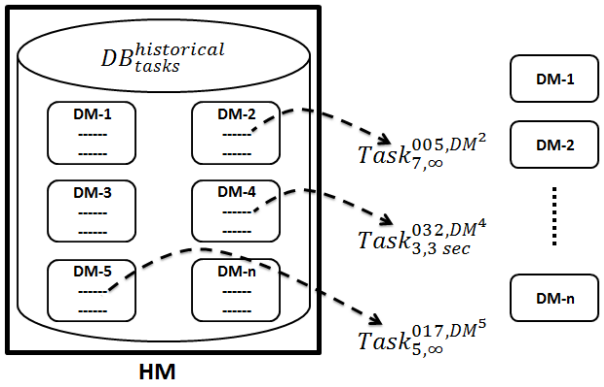


FIGURE 11. DB of historical tasks.

In Figure 11, $Task_{7,\,\infty}^{005,\,DM^2}$ and $Task_{5,\,\infty}^{017,\,DM^5}$ refer to two normal tasks that are executed at DM^2 and DM^5, and, in the past, they were executed seven and five times, respectively. Task $Task_{3,\,3\,\mathrm{sec}}^{032,\,DM^4}$ is an *TST* executed at DM^4 three times in the past, and $dl = 3$ sec.

Definition 3: For a given number of tasks (α) that are executed at the i^{th} DM and recorded in $DB_{tasks}^{historical}$, each task has an execution probability denoted as (*ExeP*). Then, from a mathematical point of view,

$$\sum_{j=1}^{\alpha} ExeP_j = 1 \tag{4}$$

According to the i^{th} DM, the tasks differ or may be equal in terms of *ExeP*, as shown in Figure 12.

In Figure 12, the historical tasks executed at DM-2 have different execution probabilities, which are represented by the separated column. For example, among the α tasks, there are 18 tasks coloured yellow, and each task has the same query probability (0.0028).

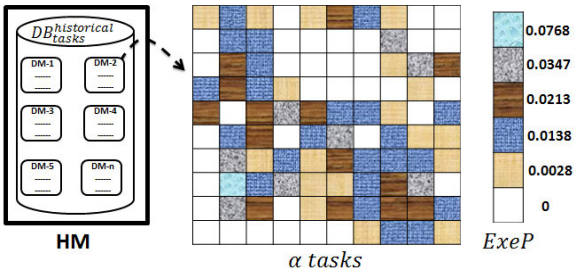


FIGURE 12. Tasks executed at DM-2 with execution probabilities.

Definition 4: (γ − *anonymity*) refers to the anonymity set that protects the real task. Formally, for the γ tasks, executed at the i^{th} DM, which includes a real task and ($\gamma - 1$) dummy tasks, each task has a conditional probability of being a real executing task. Let $ExeP_j\,(j = 1, 2, \ldots, \gamma)$ denote the probability that the j^{th} task is the real executing task; then,

$$ExeP_j = \frac{ExeP_j}{\sum_{j=1}^{\gamma} ExeP_j} \tag{5}$$

According to Shannon's principle [29], stemming from information theory, the entropy (ε) of identifying the real task out of the anonymous set is given as

$$\varepsilon = -\sum_{j=1}^{\gamma} ExeP_j \times \log_2(ExeP_j) \tag{6}$$

Definition 5: (RaS_{tasks}^{dummy}) refers to a set of ($\gamma - 1$) dummy tasks that are selected from the α tasks that were executed at the i^{th} DM. Formally, this is given as

$$RaS_{tasks}^{dummy} = \left\{ \begin{array}{l} Dtask_1^{\,Task_{freq,dl}^{ref,DM^i}} \\ Dtask_2^{\,Task_{freq,dl}^{ref,DM^i}} \\ \ldots\ldots \\ Dtask_{\gamma-1}^{\,Task_{freq,dl}^{ref,DM^i}} \end{array} \right\} \tag{7}$$

The DTS approach forms the RaS_{tasks}^{dummy}, where it selects the tasks randomly without further considerations. For a real task ($ReTask_{freq,\,dl}^{ref\,,\,DM^i}$) that is intended to be executed at the i^{th} DM, the DTS approach selects (7) dummy tasks to achieve (8 − *anonymity*) protection level, as shown in Figure 13 ($DM^i = DM^2$).

As shown in Figure 13, the real task (coloured by green) falls in a yellow cell, which means that its execution probability is 0.0028. The DTS randomly selects seven dummy tasks (coloured by red), where none of the dummy tasks have the same execution probability as the real one.

When the mobile agent migrates to a malicious DM (attacker), executing the anonymous set there in parallel,

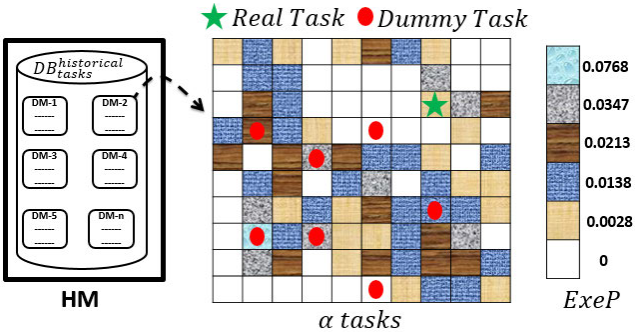


FIGURE 13. Generating dummy tasks by the DTS approach.

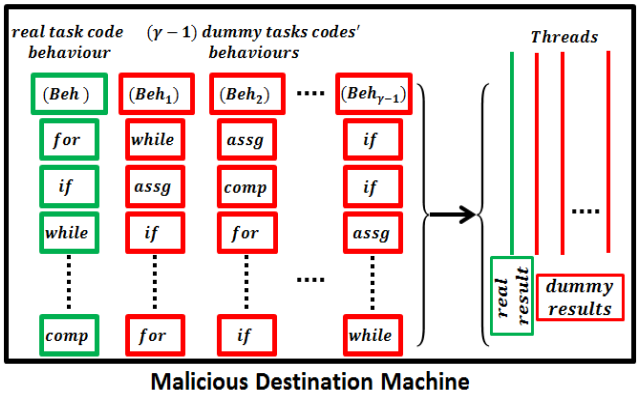


FIGURE 14. Ensuring the integrity of both the execution and results by the DTS approach.

the attacker cannot recognize the real task among the dummies. This means that the execution integrity is protected since the attacker cannot distinguish the behavior of the real task. This in turn leads to the generated real result being surrounded by dummy results, and the attacker cannot determine the correct results for modification (i.e., the result integrity is ensured). Figure 14 illustrates this idea.

Algorithm 1 shows the steps of the DTS approach.

Algorithm 1 Dummy Task Selection

Input: execution probabilities $ExeP$, γ value, real task $\mathrm{Re}Task_{freq,\,dl}^{ref,\,DM^i}$

Output: dummy task set (RaS_{tasks}^{dummy})

$RaS_{tasks}^{dummy} = \phi$
for$(j = 1; j \leq \gamma - 1, j++)$ **do**
  begin
    randomly select $Dtask_j \mid Dtask_j \in DM^i$
    add $Dtask_j$ to the RaS_{tasks}^{dummy}
end
output: RaS_{tasks}^{dummy}

Since the attacker possesses the side information mentioned in the threat model above, the frequency of execution of each task can be employed to infer the real task as well as apply some analysis to the executing tasks. To avoid this and to achieve greater defenses against malicious DMs, we enhance the DTS approach, as explained below.

2) IMPROVED-DUMMY TASK SELECTION (IMPROVED-DTS) APPROACH

The key idea of enhancement is to consider two factors: (1) selecting the dummies so that the execution probabilities of each dummy are the same as the real task and (2) selecting the dummies that have the same type of real task (i.e., normal or time-sensitive task).

For the first factor, since a larger entropy value leads to a higher uncertainty in identifying the real task from the dummy tasks, our goal is to achieve the highest entropy value. The maximum entropy value is achieved when all $(\gamma - 1)$ dummy tasks are treated as the real task (i.e., all of them have the same execution probability). The maximum entropy value is given as

$$\varepsilon_{\max} = \log_2(\gamma) \tag{8}$$

Definition 6: (CaS_{tasks}^{dummy}) refers to a set of (β) candidate dummy tasks that have the same execution probability as the real task (regardless of the type of task). From a mathematical point of view, forming the CaS_{tasks}^{dummy} set is considered an objective optimization problem. It is generated from the α tasks that were previously executed at the i^{th} DM, where $\beta \leq \alpha$. Thus, the candidate dummy tasks are given by

$$CaS_{tasks}^{dummy} = \arg\max \left\{ -\sum_{j=1}^{\beta} ExeP_j \times \log_2(ExeP_j) \right\} \tag{9}$$

Figure 15 illustrates building the candidate dummy tasks based on the maximum entropy.

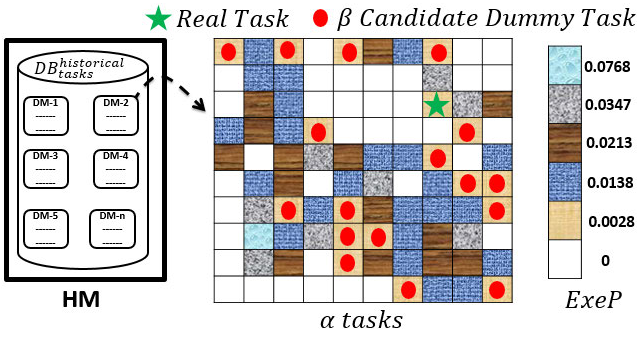


FIGURE 15. Generating the candidate dummy set by the improved DTS approach.

As shown in Figure 15, the real task (coloured by green) falls in a yellow cell, which means that its execution probability is 0.0028. The improved DTS forms the candidate of the dummy set by selecting all the tasks that fall in yellow cells ($\beta = 17$ dummy task).

Since each task included in the CaS_{tasks}^{dummy} set is of the normal task type or time-sensitive task type, the CaS_{tasks}^{dummy} set must be filtered to obtain only the tasks that agree with

the type of the real task. This means that the second factor is considered in the improved DTS approach.

Definition 7: ($SCaS_{tasks}^{dummy}$) refers to a second set of (δ) candidate dummy tasks that have the same execution probability as well as the same task type as the real task. For a given CaS_{tasks}^{dummy} set, the $SCaS_{tasks}^{dummy}$ set is formed by selecting δ candidate dummy tasks out of the β candidate dummy tasks based on the value of the deadline (dl) of the real task. Formally, for a given real task ($ReTask_{freq,\,dl}^{ref,\,DM^i}$), $SCaS_{tasks}^{dummy}$ is defined as

$$SCaS_{tasks}^{dummy} = \bigcup_{k=1}^{\delta} (Task_{freq,\,dl_k}^{ref,\,DM^i} \in CaS_{tasks}^{dummy}) \mid dl_k = dl_{real} \tag{10}$$

Figure 16 illustrates building the second candidate dummy tasks based on the type of real task.

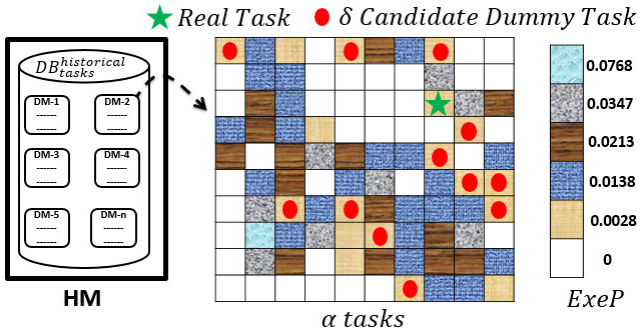


FIGURE 16. Generating the second candidate dummy set by the improved DTS approach.

In Figure 16, among the tasks ($\beta = 17$ dummy task) located in Figure 15, five cells are filtered because they do not match the type of the real task. The improved DTS forms the second candidate of the dummy set by selecting the tasks that fall in yellow cells whose type also matches the type of the real task ($\delta = \beta - 5 = 17–5 = 12$ dummy tasks).

Definition 8: (FS_{tasks}^{dummy}) refers to the final set of dummy tasks that will be executed at the malicious DM along with the real task. To achieve a desired protection level (i.e., $\gamma - 1$ *anonymity* level), the FS_{tasks}^{dummy} set is generated by randomly selecting the ($\gamma - 1$) dummy task from the $SCaS_{tasks}^{dummy}$ set. This is defined as

$$FS_{tasks}^{dummy} = \bigcup_{d=1}^{\gamma-1} rand\ (Task_{freq,\,dl_d}^{ref,\,DM^i} \in SCaS_{tasks}^{dummy}) \tag{11}$$

After forming the final set of dummy tasks, they will be executed at the DM. According to this execution event, the execution probability of each task involved in the execution event changes (i.e., the tasks included in the final set of dummy tasks). Due to this change, the execution probabilities of the tasks that are not selected to be involved in the execution event also change. In other words, the execution probability of all tasks changes dynamically with the execution of each task. Depending on the frequency of the execution of the task ($freq$), we model this issue mathematically.

Originally, for a given task executed previously ($freq = n$) in the second destination machine $DM^{i=2}$, which contains α tasks, the set $ExeP$ is defined according to its frequency as

$$ExeP(task) = \frac{n}{\sum_{k=1}^{\alpha} freq_k} \tag{12}$$

Let $cur(freq)$ and $fut(freq)$ refer to the current frequency and the future (after execution event) frequency of a task, respectively. The future frequency for each task is defined as

$$fut(freq) = \begin{cases} cur(freq) + 1, & task \in FS_{tasks}^{dummy} \\ cur(freq) + 0, & task \notin FS_{tasks}^{dummy} \end{cases} \tag{13}$$

For generalization, the execution event causes a change in the execution probability of other tasks. Let $ExeP_{task}^{fut}$ refer to the future execution probability for a task. Then, it is defined based on both the $fut(freq)$ and the level of desired anonymity (γ) as

$$ExeP_{task}^{fut} = \frac{fut(freq)}{\sum_{k=1}^{\alpha} fut(freq) + \gamma} \tag{14}$$

Security Gap and Dummy Times: The improved DTS approach has a security gap against DoS attacks. The attacker can attempt to modify the deadlines of all the tasks (i.e., the real and dummy tasks), making their execution time higher than the longest deadline found in FS_{tasks}^{dummy}. Using this method, the attacker guarantees that the real task will definitely exceed its own deadline. To solve this problem, we rely on the dummy term to regenerate the deadlines of the tasks. The key idea is to make all deadlines of the tasks uniform (the real tasks and those included in the FS_{tasks}^{dummy} set). This is performed according to the deadline of the real task so that the uniform deadline will be equal to half of the original deadline of the real task. Using dummy times (uniform deadline) ensures robustness against the DoS attack because the deadline of the real task cannot be recognized among the dummy times. However, setting the uniform deadline as half of the original deadline of the real task can be used as a second level of protection against the DoS attack. In other words, even if the attacker updated the uniform deadline by duplicating it to enable the success of the DoS attack, this malicious trail fails because the original deadline of the real task will not exceed the modified deadline performed by the attacker side.

Algorithm 2 shows the steps of the improved DTS approach.

C. THEORETICAL ANALYSIS ON THE TIME COMPLEXITY

After sorting the tasks based on their query probability, the algorithm needs to choose candidate tasks whose history execution probabilities are similar to those of the real task. In this work, we set the size of the CaS_{tasks}^{dummy} to be double the

Algorithm 2 Improved-Dummy Task Selection

Input: execution probabilities $ExeP$, γ value, real task $\mathrm{Re}Task^{ref,DM^i}_{freq,dl}$

Output: final dummy task set FS^{dummy}_{tasks}

$CaS^{dummy}_{tasks} = SCaS^{dummy}_{tasks} = FS^{dummy}_{tasks} = \phi$
sort tasks based on their execution probability
while $(DB^{historical}_{tasks} \neq \phi)$ **do**
begin
  select $Dtsk_j \mid (Dtsk_j \in DM^i$ and $ExeP(Dtsk_j) = ExeP(\mathrm{Re}Task))$
  add $Dtsk_j$ to the CaS^{dummy}_{tasks}
end
while $CaS^{dummy}_{tasks} \neq \phi$ **do**
begin
  select $Dtsk_j \mid type(Dtsk_j) = type(\mathrm{Re}task)$
  add $Dtsk_j$ to the $SCaS^{dummy}_{tasks}$
end
for $(k = 1; k \leq \gamma - 1, k++)$ **do**
begin
  randomly select $Dtsk_k \mid Dtsk_k \in CaS^{dummy}_{tasks}$
  add $Dtsk_k$ to the FS^{dummy}_{tasks}
end
output: FS^{dummy}_{tasks}

required level of security (i.e., the improved-DTS approach chooses 2γ candidate tasks). In the 2γ candidate tasks, $\gamma - 1$ tasks are randomly selected. Then, the approach derives ∂ sets, and each set contains γ tasks. For each set, one task is real, and the other $\gamma - 1$ tasks are randomly chosen from the 2γ candidates. In this context, $\partial = C^{\gamma-1}_{2\gamma}$, and the entropy for the $j^{th}(j \in [1, \partial])$ set can be calculated according to Equation (6). It is obvious that the greater the value of ∂ is, the higher the computational cost of the improved DTS algorithm is. In addition, different values of ∂ may result in different optimal task sets in the improved-DTS algorithm, and the improved-DTS algorithm can obtain the optimal location set when $\partial = C^{\gamma-1}_{2\gamma}$. In terms of time complexity and using the notations summarized in Table 3, we analyze the performance of the improved DTS algorithm when $\partial = C^{\gamma-1}_{2\gamma}$ as follows.

1) BEST PERFORMANCE

$\exists i, j \in [1, N], ExeP_i = ExeP_j (i \neq j)$ is the set $ExeP$. We assume that for a particular task i the number of tasks whose execution probabilities are the same as that of the real task in the chosen candidate tasks is denoted by $\hat{\gamma}$. Since $ExeP_i = ExeP_j$, the set $ExeP_i$, the procedure selected for task i, is the same as the set $ExeP_j$, the procedure selected for task j, in the improved-DTS algorithm. We then discuss the performance of the improved DTS algorithm in the following situations. When $1 \leq \hat{\gamma} \leq \gamma - 1$, set C_i as the same as C_j in the improved-DTS algorithm under the condition $\partial = C^{\gamma-1}_{2\gamma}$. In this situation, although the DM (attacker) can infer the probability for a task selection, the DM cannot know the real task. This phenomenon is observed because there are other tasks whose execution probabilities are the same as that of the real task. Moreover, the larger $\hat{\gamma}$ is, the better the performance of the improved DTS algorithm is. When $\hat{\gamma} \geq \gamma$, C_i may be different from C_j. This difference is observed because randomly selecting $\gamma - 1$ tasks from the $\hat{\gamma}$ tasks whose execution probabilities are the same as that of $ExeP_i$ may yield the optimal task set. In this situation, since each task has the same execution probability, the improved-DTS algorithm achieves the best performance.

TABLE 3. Key notations.

Notation	Description
N	Number of all tasks.
γ	The required security level.
$ExeP[N]$	The historical execution probabilities for all tasks.
∂	Number of randomly selecting $\gamma-1$ tasks from 2γ tasks (i.e., $\partial = C^{\gamma-1}_{2\gamma}$).
$ExeP_i$	The historical execution probability for task i.
$ExeP_i[2\gamma]$	The chosen 2γ candidates for task i.
$C_i[\gamma]$	The chosen optimal task set for task i.
$\hat{\gamma}$	The number of tasks which have the same historical execution probability as the real task in $ExeP_i$.

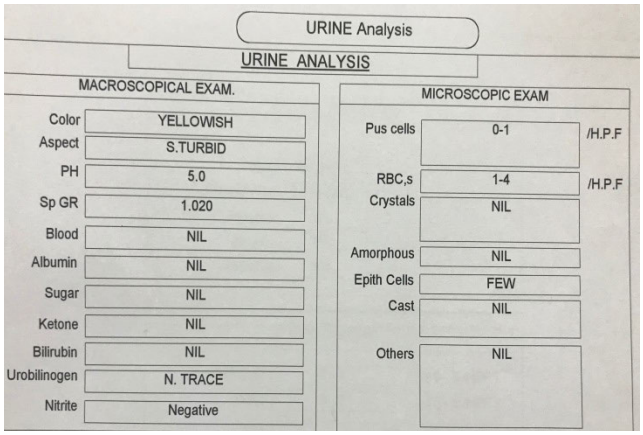
2) BAD PERFORMANCE

$\exists i, j \in [1, N], ExeP_i \neq ExeP_j (i \neq j)$ is the set $ExeP$. Since $ExeP_i \neq ExeP_j$, the set $ExeP_i$, the procedure selected for task i, must be different from the set $ExeP_j$, the procedure selected for task j. However, when $\partial \leq \gamma - 1$, C_i may be the same as C_j, that is, the chosen optimal task set for task i is likely to be the same as the chosen optimal task set for task j. In this situation, although the DM may try to infer which task is most likely to select this task set, the DM may make an incorrect decision. The reason is that the optimal task sets chosen by the procedure in other tasks are the same as that of the real task. Moreover, the larger the number of tasks whose chosen optimal task sets are the same as those of the real task is, the better the performance of the improved DTS algorithm is. However, if there is no task whose chosen optimal task set is the same as that of the other tasks in the set $ExeP$, then the improved-DTS algorithm will have bad performance.

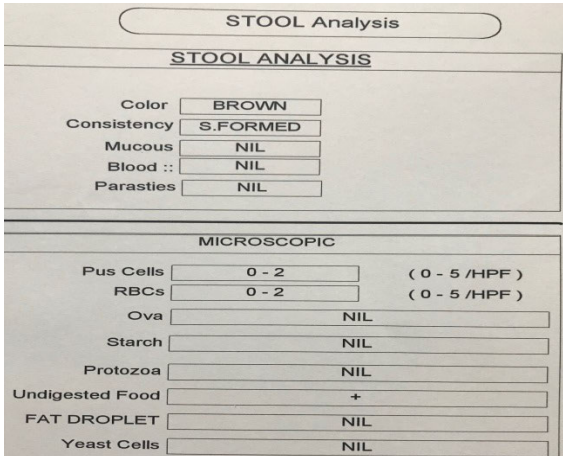
D. A SPECIFIC APPLICATION EXAMPLE OF DTS AND IMPROVED-DTS APPPROACHES

To demonstrate a real application of the DTS and improved DTS, we present a specific example of the DTS and the improved-DTS in the medical domain. The reason for selecting the medical domain is that it contains both normal

and time-sensitive tasks, which are suitable for a historical database description. The patients whose medical cases are critical are admitted to the intensive care room. The biological signs of the bodies of such patients need to be monitored throughout the period of medical care. There are important medical signs that indicate the status of the body's vital (life-sustaining) functions, these include body temperature (BT), blood pressure (BP), heart rate (HR), and breathing rate (BR). Such vital functions represent medical time-sensitive tasks and are required to be monitored periodically because a poor status of a vital function reflects a bad indicator (i.e., the life of the patient is threatened). Other functions are considered normal medical tasks. Examples of normal medical tasks include urine analyses and stool analyses. The following figure shows real medical analyses (normal medical tasks).



(a) Urine analysis.



(b) Stool analysis.

FIGURE 17. Examples of real medical tests.

In Figure 17, there are 30 medical tasks (18 urine analysis tasks and 13 stool analysis tasks). The result of each task can be obtained using a mobile agent, for example, the level of acidity in the urine (i.e., PH = 5 in the urine analysis).

In Figure 18 below, we arranged 26 normal medical tasks similar to those shown in Figure 17 along with 4 time-sensitive medical tasks in two tables that show the medical

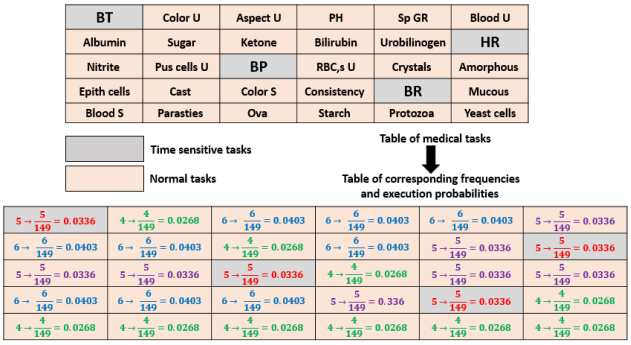


FIGURE 18. Corresponding medical task frequencies and execution probabilities.

tasks with the corresponding frequency and execution probability of each task.

Using the information in Figure 18, the DTS approach selects dummy tasks randomly without taking into consideration any factor in the selection process. In contrast, the improved-DTS approach selects dummy tasks while considering two factors (execution probability and type of the task). Suppose that we create a mobile agent with a task related to finding the pH value of the urine, and the required security level is 5 (i.e., anonymity level is $\gamma - 1 = 4$). During execution, the DTS approach selects, for example, the following dummy tasks: BP, Ova, Colour S, and Nitrite, with the corresponding execution probabilities: 0.0336, 0.0268, 0.0403, and 0.0336, respectively. Based on the selected dummies, the entropy value is

$$\begin{aligned}\varepsilon = &-(0.0336 \times \log(0.0336)) - (0.0268 \times \log(0.0268)) \\ &-(0.0403 \times \log(0.0403)) - (0.0336 \times \log(0.0336)) \\ &-(0.0403 \times \log(0.0403)) = 0.584\end{aligned}$$

Since the improved-DTS selects the dummies such that each dummy has an execution probability equal to that of the real task, where any four tasks among the nine tasks available in the medical data set can be chosen, the entropy value is

$$\begin{aligned}\varepsilon = &-(0.0403 \times \log(0.0403)) - (0.0403 \times \log(0.0403)) \\ &-(0.0403 \times \log(0.0403)) - (0.0403 \times \log(0.0403)) \\ &-(0.0403 \times \log(0.0403)) = 0.0403 = 0.648\end{aligned}$$

By comparing the results generated by the DTS and improved-DTS approaches, we can infer that the improved-DTS is stronger than the DTS in terms of entropy.

The lower bound of the number of tasks in the historical task database. It is worth mentioning that the level of danger is high when only one existing task matches the criteria of the improved-DTS approach. This is because the probability of determining the real task as opposed to a dummy task is 50 %, which is notably high. In addition, it is obvious that as the number of dummy tasks increases, the probability of recognizing the real task decreases. This leads to the creation of a lower bound on the number of tasks in the historical

task database. Since the final set of dummy tasks is formed by randomly selecting the dummy tasks from the second candidate dummy set, then

$$size(SCaS_{tasks}^{dummy}) \rangle size(FS_{tasks}^{dummy}) \tag{15}$$

where

$$size(FS_{tasks}^{dummy}) = \gamma - 1 \tag{16}$$

In terms of the sizes, the lower bound ($Bound_{lower}$) of the number of tasks in the historical task database is determined based on the following condition:

$$size(SCaS_{tasks}^{dummy}) \rangle Bound_{lower} \geq \gamma \tag{17}$$

In the case where the condition on the lower bound is not satisfied, it is essential to generate additional imaginary dummy tasks. This case will be considered in future work.

IV. SECURITY ANALYSIS

In this section, we discuss the resistance of the improved DTS approach against the attacks included in the threat model above. In addition, we address the process of reversing the algorithm at the attacker side.

A. RESISTANCE AGAINST ATTACKS

The information possessed by the attacker includes the execution probability ($ExeP$) of each individual task and all γ tasks (i.e., the $\gamma - 1$ dummy tasks included in the FS_{tasks}^{dummy} set as well as the real task). The key to the success of alternation, collusion, and DoS attacks is to determine the real task successfully, which is followed by malicious actions. Let P_{Guess}^{e} denote the probability that the attacker can successfully guess if the event e is true. Then, an approach is resistant to attacks if

$$P_{Guess}^{e_1=Task_i \in FS_{tasks}^{dummy}} = P_{Guess}^{e_2=Task_j \in FS_{tasks}^{dummy}} \quad \forall (0 < i \neq j \leq \gamma) \tag{18}$$

Proof: The attacker cannot obtain any benefit from employing the $ExeP$ to determine the real task. This is because the process of selecting the dummy tasks depends on the same values of the $ExeP$. This problem (i.e., determining the real task among the dummies) will be accentuated at the attacker's side since all tasks are of the same type. Therefore, the probability of successfully guessing the real task is $\frac{1}{\gamma}$, where this value is the same for all executed tasks at the attacker's side. Thus, the only choice for the attacker is to guess the real task in a random manner. Guessing the real task randomly means being unable to identify the real result (generated by the real task) among the dummy results (generated by the dummy tasks), which in turn means that the attacks fail. It is worth mentioning that even if the attacker attempts to apply some time-based analysis, no benefit is gained since all tasks have the same deadline.

B. REVERSING THE ALGORITHM

We assume that the attacker knows the proposed algorithm of the improved DTS approach; therefore, he/she may attempt to reverse the algorithm, but this will fail. This is because the content of the FS_{tasks}^{dummy} set is generated randomly. This randomization used to construct the actual dummy tasks guarantees the uncertainty of the selection, which, in turn, leads to uncertain dummy selection results. As a result, even if the attacker runs our proposed algorithm several times, he/she cannot infer the real task and, of course, the real corresponding result.

C. SPECIAL CASE

In the case of an existing DM, the tasks in the HM that are collected are almost the same as the tasks in the historical task database, and the probability of the DM identifying the real task depends on pure random guessing. This is because the content of the FS_{tasks}^{dummy} set is generated randomly. This randomization used to construct the actual dummy tasks guarantees the uncertainty of the selection, which, in turn, leads to uncertain dummy selection results. Mathematically, the probability of identifying the real task is

$$pro = \frac{1}{size(dataset)} \tag{19}$$

where $size(dataset)$ refers to the number of tasks located at the DM. As a result, the method is considered safe from a mathematical perspective.

V. USED METRICS

In this section, we provide the metrics that are used for evaluation purposes. Two types of metrics are used in this work: security metrics and performance metrics.

A. SECURITY METRICS

We consider the user, who wants to protect the sending agent (or the agent's own side), as well as the attacker's side.

For the agent's side, since the entropy ε measures the uncertainty of determining the real task among the dummy tasks, we employ it as a security metric to quantify the protection level that is achieved. A higher entropy value means a higher security protection level and vice versa. The entropy is identified by equation 6 above.

For the attacker's side, we define the level of violation (LoV) metric. The LoV metric is derived from the entropy as follows. The highest entropy value (i.e., the optimal value) that can be achieved is $\varepsilon_{\max} = \log_2(\gamma)$, which is achieved when all the executed dummy tasks have the same probability of being treated as the real task. If the agent's owner achieves an entropy value ($\hat{\varepsilon}$) that is less than $\varepsilon_{\max}$, then the attacker succeeds in violating the defense by

$$LoV = \log_2(\gamma) - \hat{\varepsilon} \tag{20}$$

Figure 19 illustrates the key idea of the level of violation metric.

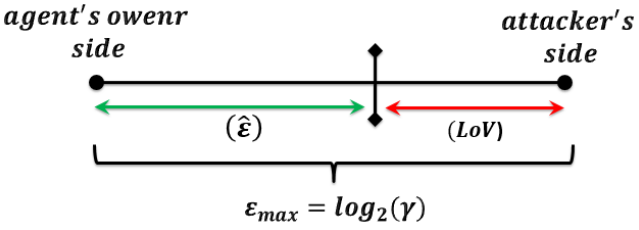


FIGURE 19. Level of violation LoV metric.

It is obvious that a higher LoV value means a lower resistance against attacks and vice versa.

B. PERFORMANCE METRICS

We measure the performance in terms of time. In this context, we consider the processing time spent in both the home machine and destination machine. We define the following times:

1) T_{proc}^{HM} denotes the time spent at the HM, including the time of creating the agent, time of defining the itinerary, and any other time needed for the protection mechanism.
2) T_{proc}^{DM} denotes the time spent at the DM, including the time needed for the protection mechanism and the delay spent starting execution.
3) T_{exe}^{task} denotes the time spent at the DM to execute the task performed by the agent.
4) T_{sent}^{agent}
5) denotes the time required to send the agent to the DM (i.e., migration).
6) $T_{recieve}^{agent}$ denotes the time required to receive the agent at the HM (i.e., migration back).

Then, the total time required from the moment of creating the agent to the moment of receiving the results is given by

$$T_{total}^{agent} = T_{proc}^{HM} + T_{proc}^{agent} + T_{proc}^{DM} + T_{exe}^{task} + T_{recieve}^{agent} \quad (21)$$

To generalize the total time to be suitable for other protection mechanisms, we ignore the T_{sent}^{agent}, T_{exe}^{task}, and $T_{recieve}^{agent}$ times because they are not related the protection mechanism being applied. In other words, we include only the time required to perform the protection mechanism at the HM and the DM. Thus, the total time will be as follows:

$$T_{total}^{agent} = T_{proc}^{HM} + T_{proc}^{DM} \quad (22)$$

In practice, as the total time decreases, a better performance is achieved.

VI. EXPERIMENTAL RESULTS AND EVALUATIONS

We implement the proposed system on the Aziz supercomputer available at King Abdulaziz University, Jeddah Saudi Arabia, to enable parallel execution. The Aziz supercomputer has the specifications summarized in the following table.

An artificial medical database, which consists of 10,000 records, is used. Each record represents a task that contains the following fields:

TABLE 4. Key notations.

Specification	Value
Number of nodes	380
Number of cores	9120
Memory GB	96
GPU nodes cores	4992 CUDA
Processors	Intel Xeon Phi

1) *freq* represents the frequency of the execution of the task.
2) *ref* represents the ID of the task.
3) *dl* represents the deadline of the task and is used to determine whether the task is a normal task or a time-sensitive task.
4) DM_i represents the destination machine where the task is executed.
5) *ExeP* represents the execution probability of the task.

The normal tasks are related to performing certain queries regarding a diagnosis, collecting information, or reporting the cases of the patients.

Time-sensitive tasks are related to critical medical situations that threaten the lives of patients and require immediate information about such measurements as heartbeat, blood pressure, and temperature. More than 1000 tasks are executed and stored in the database as historical tasks. In addition, TSTs are performed, therein exploiting the multiple GPUs contained in the Aziz supercomputer. The TSTs used in the system are inspired by ref [19] for adaptation in the medical area. The execution probabilities are generated randomly and associated with the tasks.

We select the fragmentation-based encryption (FBE) and the obfuscated code (OC) approaches for comparison purposes with the DTS and the improved-DTS approaches in terms of dummy sets. This is because the code generated by the OC-based approach can be considered as a dummy for the real code, the same as with the fragmented code generated by the FBE-based approach.

1) ENTROPY-BASED EVALUATION

Figure 20 shows the relationship between γ and the security level.

Generally, the entropy increases as γ increases. This is reflected in Figure 18 by both the DTS and improved-DTS approaches. However, the improved-DTS approach performs the best, overcoming the DTS approach, because it is designed to achieve the highest entropy values based on the same execution probability values. The DTS approach does not consider the previous factor, where it selects the dummy tasks randomly.

For the FBE and OC approaches, γ is limited to 2 because each approach generates only one task code as a dummy of the real task code. Both approaches retain the same value

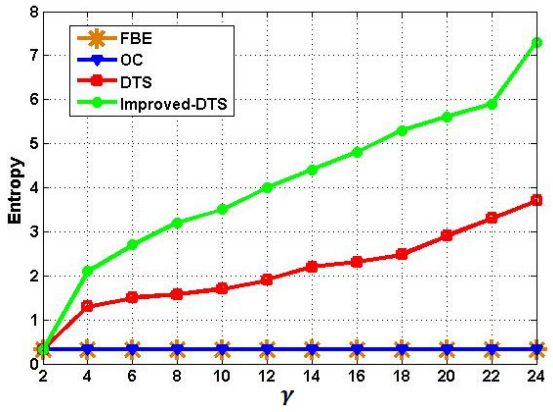


FIGURE 20. Entropy vs γ.

regardless of the increased γ value because the execution probability value of the dummy task will be the same as the real task. Therefore, the FBE, OC, and improved DTS approaches achieved the same entropy value when $\gamma = 2$. However, the DTS approach performs slightly poorer than the other approaches when $\gamma = 2$ because the randomly selected dummy task has a low execution probability compared to the real task in the conducted experiment.

2) RESISTANCE IN TWO ATTACKS

In this context, we measure the robustness of the approaches involved in the comparison against a mixture of alternation and collusion threats. We use the LoV metric to measure the resistance, where we define a threshold ($LoV = 0.75$) above which the agent is considered at the danger level. A total of 20 different tasks are used with a security level ($\gamma = 6$), as shown in Figure 21.

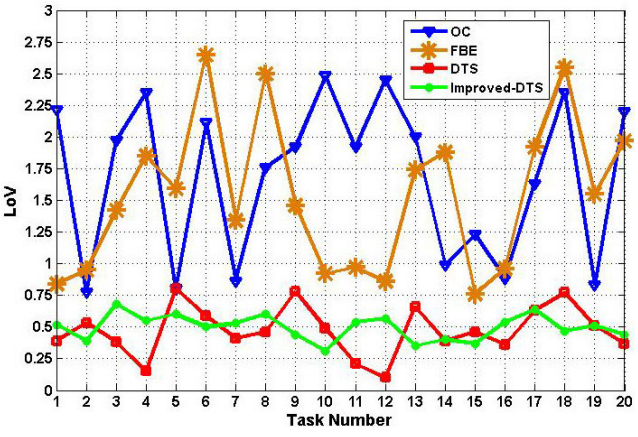


FIGURE 21. *LoV* value for 20 tasks, $\gamma = 6$, *threshold* $= 0.75$.

As shown in Figure 19, most of the tasks that are performed by the mobile agents and protected based on the FBE and OC approaches are attacked by the destination machine. This is mainly because these approaches can generate only one dummy for each real task. The DTS approach outperforms the OC and FBE approaches. This is because of the higher value of the entropy, given that it generates five dummies. This leads to most of the agents being safe and the corresponding tasks not being attacked by the destination machine. However, some agents were on the danger side since the approach ignores the fact that the attacker may employ side information (execution probability) for malicious purposes. As a result, some of the chosen dummies may present a very low execution probability and consequently are filtered by the attacker. The improved-DTS approach performs the best, where all agents exceeded the danger threshold. This is because the approach generates five strong dummy tasks based on both execution probability and type of task. Thus, the improved-DTS approach has the highest resistance against a mixture of both alternation and collusion attacks. Compared to the improved-DTS, the DTS approach sometimes perform better. This phenomenon is attributable to the pure-random selection in the DTS. It happens that by chance, the execution probability of each selected dummy task is the same as that of the real task, in addition, the dummy and real tasks are of the same type. This is because the probability of each dummy task being of the same type as the real task is high (50 %) because we have only two choices (normal task or time sensitive task). In such cases, if the execution probability of the real task is originally high, the value of the LoV will be lower compared to the corresponding value when applying the improved DTS. Table 5 summarizes the results obtained from Figure 19.

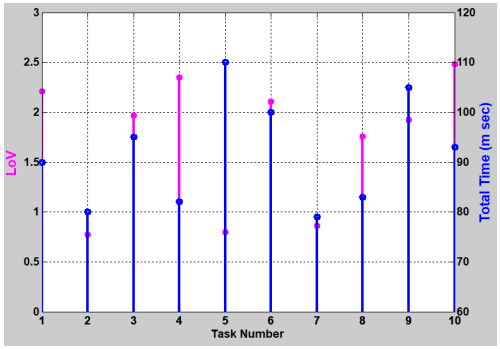
TABLE 5. Resistance of approaches.

Settings: $\gamma = 6$, $threshold = 0.75$		
Approach	Number of tasks that exceeded threshold	Percentage of violation
Improved-DTS	0	0
DTS	3	0.15
OC	20	100
FBF	20	100

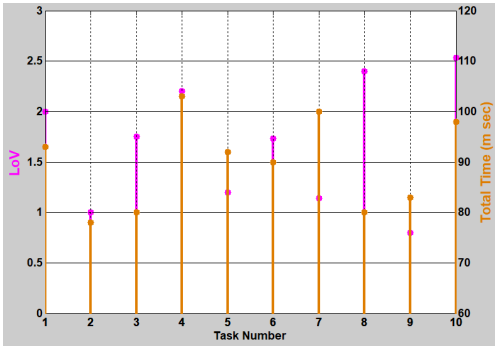
TABLE 6. Resistance-of-violation-based comparison.

Exp NO	NO of tasks	Threshold value	% of violation predefined threshold			
			FBF	OC	DTS	Improved -DTS
1	35	0.7	100	100	0.15	0
2	50	0.56	100	100	0.18	0
3	65	0.6	100	100	0.22	0
4	80	0.55	100	100	0.23	0
5	95	0.5	100	100	0.27	0.1

Note that the resistance of both the OC and FBE approaches depends on the value of the execution probability of the current task. Therefore, the OC approach sometimes outperforms the FBE approach and vice versa. We re-execute

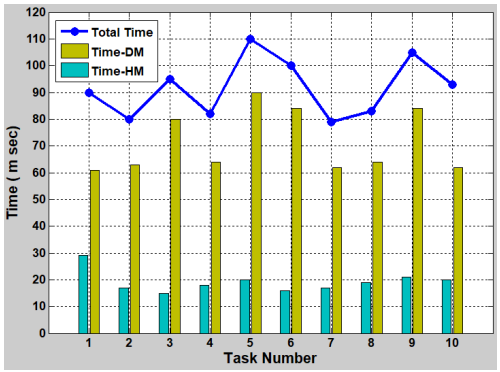


(a) OC approach.

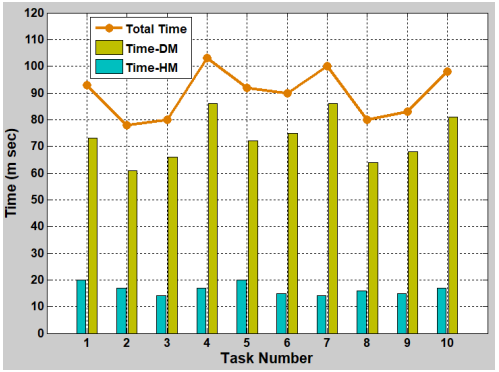


(b) FBE approach.

FIGURE 22. Resistance under mix of three attack methods, $LoV = 0.75$, $dl = 60\,m$ sec.



(a) OC approach.



(b) FBE approach.

FIGURE 23. Total time analyses.

the experiments under different decreased threshold values and different numbers of tasks, as shown in Table 6.

The results of Table 6 support those found in Table 5, where both the FBE and OC approaches show the lowest resistance against attacks, with 100% violation. Compared to the improved-DTS approach, the DTS approach performs poorly, with an increased violation percentage. The improved DTS approach shows the best resistance, with a steady violation percentage. The last violation percentage (0.1) can be justified by the success of recognizing some real tasks among dummy tasks, and this success is due to pure random guessing.

3) EVALUATION OF RESISTANCE TO THREE ATTACKS

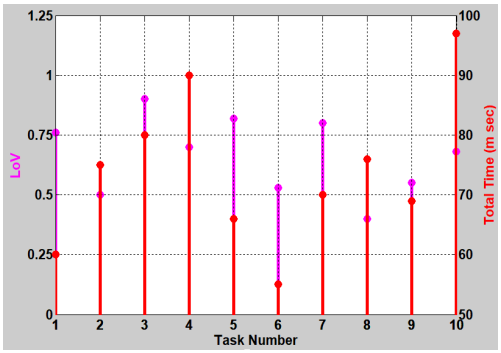
In this context, we strengthen the power of the alternation and collusion attacks by applying a DoS attack. We use both the LoV security metric and T_{total}^{agent} performance metric in this evaluation.

Ten TSTs are selected and tested under the thresholds $LoV = 0.75$ and $dl = 60\,m$ sec. Figures 22 (a) and (b) above show the resistance against the three-attack mixture for the OC and FBE approaches, respectively.

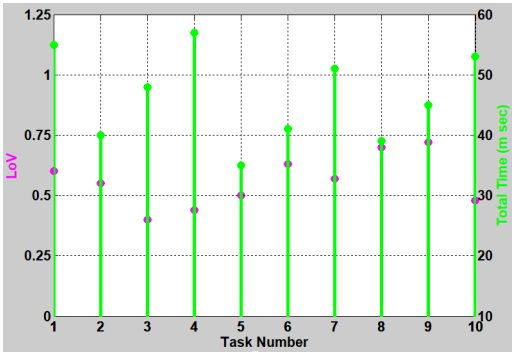
Both the OC and FBE approaches were attacked, and all tasks exceeded both the predefined threshold of the LoV and deadline. This is because the destination machines have full control over the protection mechanisms executed at their sides. In addition, due to only one dummy task being generated, the probability of determining the real task is high (50%). As a result, it is easy to tamper with the codes, results, and deadlines of the visiting agents at the attackers' side, leading to successful attacks.

To analyze the success of the DoS attack, Figure 23 above provides insight into the total time on both sides T_{proc}^{HM} and T_{proc}^{DM}.

The time spent at the HM in both the OC and FBE approaches is short compared to the time spent at the DM. This is because the process of obfuscation of the code (in the OC approach) and the process of extracting and encrypting sensitive parts of the code (in the FBE approach) take some time at the HM. In the ideal case (i.e., the DM is not malicious), it is expected to spend approximately the same amount of time reconstructing the real code (in the OC approach) and extracting and decrypting the encrypted pieces of code (in the FBE approach). However, there are two reasons behind the long time spent at the DM: (1) it is easy for the malicious DM to determine the real code, and (2) because the processes of reconstructing and decrypting code are performed in the space of the DM, the malicious DM has the ability to tamper

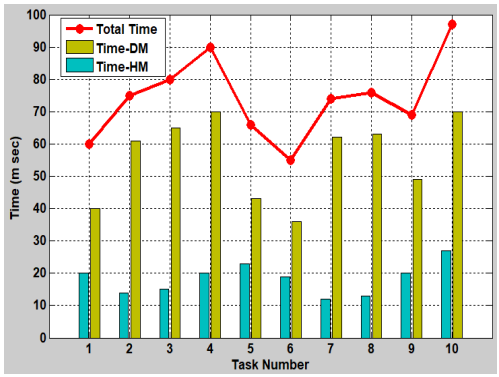


(a) DTS approach.

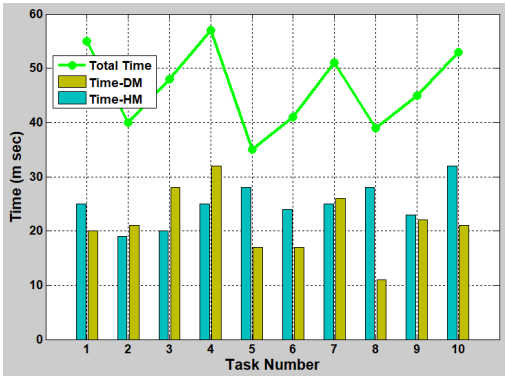


(b) Improved-DTS approach.

FIGURE 24. Resistance under mix of three attack methods, $LoV = 0.75$, $\gamma = 6$, $dl = 60\,m$ sec.



(a) DTS approach.



(b) Improved-DTS approach.

FIGURE 25. Total time analyses.

with (or lengthen) the deadlines of the tasks. This results in long delays before starting the execution of the tasks at the DM, which in turn leads to the deadlines being exceeded and thus the success of the DoS attack.

In Figure 24 above, the same ten tasks are protected using the DTS and improved-DTS approaches and are tested under the threat of three attack methods. The improved DTS approach shows better resistance to alternation and collusion attacks, where three tasks are attacked in the DTS approach. For the DoS attack, we analyze the time spent at both the HM and DM.

In Figure 25 above and for the DTS approach, the time spent at the DM is long; thus, more than half of the tasks (2nd, 3rd, 4th, 7th, 8th, and 10th tasks) exceeded the deadline, which reflects a low resistance to DoS attack. That is because the deadline was not protected, and the malicious DM has the ability to adjust the deadlines of all the tasks (including the real one) according to the longest deadline (which is originally longer than 60 ms) found in the dummy tasks regardless of whether the real task is distinguished. However, the dummy tasks contributed to protecting the four tasks against the DoS attack by confusing the malicious DM when determining the real task among the dummy tasks. Compared to the DTS approach, the time spent at the DM

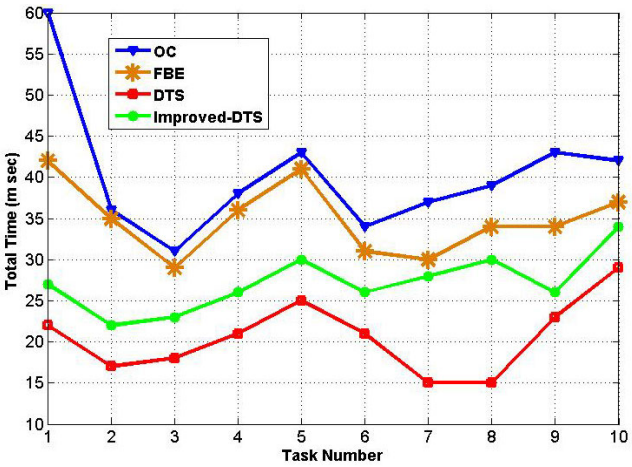
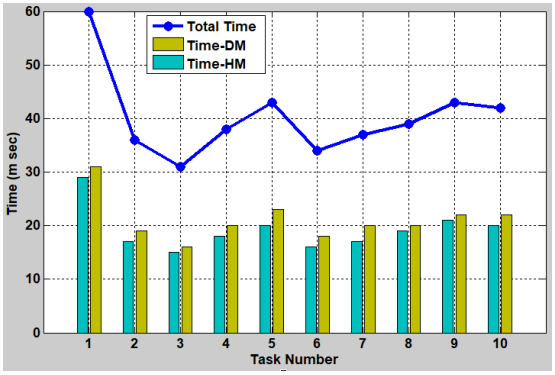
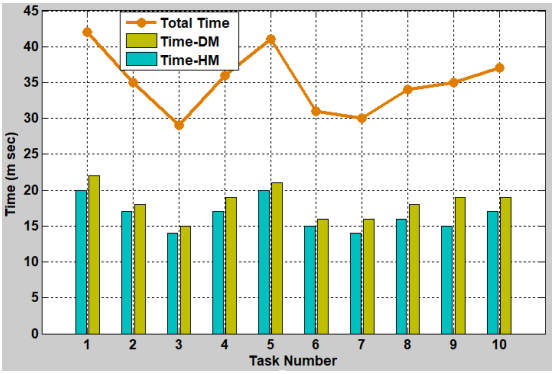


FIGURE 26. Performance of protection approaches, $\gamma = 6$.

for the improved-DTS approach is shorter, leading to a higher resistance against DoS attack (actually, none of the tasks were attacked). The underlying reasons for this are as follows: (1) the malicious DM cannot recognize the real task (to tamper with its deadline) among the dummy tasks since all tasks have the same execution probability and the same task type,

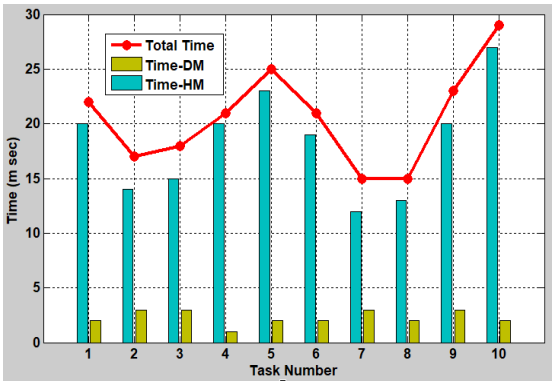


(a) OC approach.

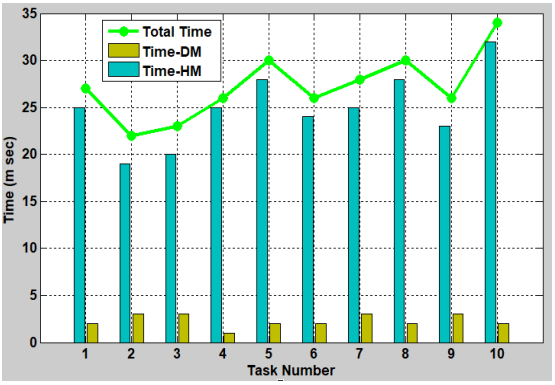


(b) FBE approach.

FIGURE 27. Total time analyses without any threat.



(a) DTS approach.



(b) Improved-DTS approach.

FIGURE 28. Total time analyses without any threat.

and (2) all tasks have uniformed deadlines, which act as dummy times to defend against DoS attacks.

Note that compared to the OC and FBE approaches, the time spent at the HM in the DTS and improved-DTS approach is longer. This in turn highlights an important feature for our proposed approaches, which is having full control over the protection mechanism at the user (owner of the agent) side. This is because from a higher security level perspective, it is preferred to apply the whole protection mechanism at the user side rather than only a part of it. This is to say that compared to performing the full protection mechanism (i.e., generating all dummies at the HM in our proposed approaches), the second part of the OC approach (reconstructing the real code) and the second part of the FBE approach (decryption of the encrypted pieces of the code) are performed at the DM, which is the attacker. This clearly justifies the higher level of protection in our proposed approaches against potential attacks.

4) PERFORMANCE-BASED EVALUATION

In this context, we evaluate the four approaches without threat of any attack. Using the T_{total}^{agent} performance metric, the same ten tasks are tested, as illustrated in Figure 26.

Without any threat, the DTS approach performs the best since it generates dummies directly without considering any other factors. Compared to the DTS, the improved DTS performs relatively poorly because it needs extra time to generate strong dummy tasks based on the execution probability, type of task, and deadline. The OC approach performs the worst because it requires a long time to obfuscate the entire original code and to reconstruct it at the DM. Compared to the OC, the FBE approach performs better because the time needed for encrypting the sensitive pieces of the original code and decrypting it at the DM requires is shorter.

Without any threat, both the OC and FBE approaches are returned to the ideal case, where the time spent at both the HM and DM is approximately the same, as shown when analyzing the total time in Figure 27 above.

The reason behind the results shown in Figure 27 is that the reconstruction of the original code from the obfuscated code requires almost the same amount of time in the OC approach. The same scenario occurs in regard to encryption and decryption processes in the FBE approach. However, the scenario changes in our proposed approaches, as illustrated in Figure 28 above.

The time spent at the DM in both the DTS and improved DTS approaches is very short compared to the time spent

at the HM. This is because all protection approaches are applied at the HM and because no other parts of the protection mechanism are required at the DM, where very little time is spent preparing the agents for execution. In other words, the execution at the DM starts directly without any delays at the DM. This in turn reflects the higher protection level in terms of time because the time spent at the DM (attacker) is very short, and, consequently, there is no chance for the attacker to perform any malicious actions on the visiting agents.

VII. CONCLUSION

In regard to managing and performing tasks over the internet, agent-based software technology (ABST) is the most popular framework. Due to the mobility feature, where an agent can migrate from the home machine to the destination machine to perform tasks, security issues are critical in this technology. The security issue in ABST becomes critical when the destination machine is the attacker, where it has full control over the visiting agent. The results of executed tasks may lose their integrity, the behavior of the agent may be changed, and advanced active attacks, such as alternation, collusion, and DoS attacks, may be applied by the attacker. In responding to this issue, we present in this paper the dummy task selection (DTS) and improved-DTS approaches. From a historical database of tasks, the DTS randomly generates dummy tasks to protect the real task, aiming at confusing the attacker when determining the real task among the dummy tasks and limiting their ability to perform malicious actions. The improved DTS aims at generating strong dummy tasks based on three factors: (1) dummy tasks have the same execution probability as the real task, which in turn guarantees the highest entropy; (2) dummy tasks are of the same type as the real task (normal or time-sensitive task); and (3) the deadlines of the dummy tasks are the same as that of the real task. Based on the entropy, total time, and level of violation (LoV) metrics and compared to well-known protection mechanisms, i.e., the obfuscation code (OC) and fragmentation-based encryption (FBE), our proposed approaches showed high resistance to advanced attacks and better performance. In regards to the resistance against alternation, collusion, and DoS attacks, the improved DTS showed the highest resistance according to the LoV security metric, where the percentage of the violation predefined thresholds was 0 and 0.1 for the first five trials and the sixth trial, respectively. In regard to performance, the DTS was ranked 1st because it performed better than the improved DTS. The root reason is that the improved DTS handles more conditions to improve the security level. In addition, considering the security issues in the context of time, our proposed approaches spend the minimum amount of time at the attacker's side.

In future work, resistance against other advanced threats, such as tailgating and blocking attacks, will be considered. In addition, there is a trade-off between the performance and the security level that is achieved. The higher the security level is, the lower the performance is. This work largely concerns the security issue. Therefore, poor performance (i.e., increasing load) can be considered out of scope. Also, a special case may be occurred if execution probability of the historical task in the database is different with the real task. In this case, creation of imaginary dummy tasks is considered in future work.

REFERENCES

[1] A. Garro, M. Mühlhäuser, A. Tundis, M. Baldoni, C. Baroglio, F. Bergenti, and P. Torroni, "Intelligent agents: Multi-agent systems," in *Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics*. 2018, p. 315.

[2] B. Alluhaybi, M. Shady, A. Alzhrani, and V. Thayananthan, "A survey: Agent-based software technology under the eyes of cyber security, security controls, attacks and challenges," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 8, pp. 1–21, 2019, doi: 10.14569/IJACSA.2019.0100828.

[3] Y. Song, A. J. Carter, J. D. Ehrlich, and S. M. Meyer, "Message passing in a distributed graph database," U.S. Patent 9 990 443, Jun. 5, 2018.

[4] R. J. Victorelli, "Remote procedure call management," U.S. Patent 10 015 283, Jul. 3, 2018.

[5] T. A. Wagner, "Predictive management of on-demand code execution," U.S. Patent 9 811 363, Nov. 7, 2017.

[6] D. B. Lange and M. Oshima, "Seven good reasons for mobile agents," *Commun. ACM*, vol. 42, no. 3, pp. 88–89, Mar. 1999.

[7] M. Karami and A. H. Shahmirzadi, "Applying agent-based technologies in complex healthcare environment," *Iranian J. Public Health*, vol. 47, no. 3, p. 458, 2018.

[8] Y. Kaeri, C. Moulin, K. Sugawara, and Y. Manabe, "Agent-based system architecture supporting remote collaboration via an Internet of multimedia things approach," *IEEE Access*, vol. 6, pp. 17067–17079, 2018.

[9] A. Caglayan and C. Harrison, *Agent Sourcebook*. Hoboken, NJ, USA: Wiley, 1997.

[10] F. Bergenti, E. Iotti, and A. Poggi, "Core features of an agent-oriented domain-specific language for JADE agents," in *Trends in Practical Applications of Scalable Multi-Agent Systems, the PAAMS Collection*. Cham, Switzerland: Springer, 2016, pp. 213–224.

[11] B. Amro, "Mobile agent systems, recent security threats and counter measures," 2014, *arXiv:1410.4147*. [Online]. Available: http://arxiv.org/abs/1410.4147

[12] F. Linna and L. Jun, "A free-roaming mobile agent security protocol against colluded truncation attack," in *Proc. 2nd Int. Conf. Educ. Technol. Comput.*, vol. 5, Jun. 2010, p. V5-261.

[13] D. Xu, L. Harn, M. Narasimhan, and J. Luo, "An improved free-roaming mobile agent security protocol against colluded truncation attacks," in *Proc. 30th Annu. Int. Comput. Softw. Appl. Conf. (COMPSAC)*, vol. 2, Sep. 2006, pp. 309–314.

[14] A. D. Wood and A. J. Stankovic, "A taxonomy for denial-of-service attacks in wireless sensor networks," in *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*. 2004, pp. 739–763.

[15] M. Min, L. Xiao, C. Xie, M. Hajimirsadeghi, and N. B. Mandayam, "Defense against advanced persistent threats in dynamic cloud storage: A colonel blotto game approach," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4250–4261, Dec. 2018.

[16] D. He, S. Chan, and M. Guizani, "Mobile application security: Malware threats and defenses," *IEEE Wireless Commun.*, vol. 22, no. 1, pp. 138–144, Feb. 2015.

[17] R. Cooley, S. Wolf, and M. Borowczak, "Secure and decentralized swarm behavior with autonomous agents for smart cities," in *Proc. IEEE Int. Smart Cities Conf. (ISC)*, Sep. 2018, pp. 1–8.

[18] B. N. Silva, M. Khan, and K. Han, "Towards sustainable smart cities: A review of trends, architectures, components, and open challenges in smart cities," *Sustain. Cities Soc.*, vol. 38, pp. 697–713, Apr. 2018.

[19] H.-Y. Wu, M. Rubinstein, E. Shih, J. Guttag, F. Durand, and W. Freeman, "Eulerian video magnification for revealing subtle changes in the world," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 1–8, Aug. 2012.

[20] N. Wadhwa, M. Rubinstein, F. Durand, and W. T. Freeman, "Phase-based video motion processing," *ACM Trans. Graph.*, vol. 32, no. 4, p. 80, 2013.

[21] M. A. Elgharib, M. Hefeeda, F. Durand, and W. T. Freeman, "Video magnification in presence of large motions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 4119–4127.

[22] B. S. Yee, "A sanctuary for mobile agents," in *Secure Internet Programming*. Berlin, Germany: Springer, 1999, pp. 261–273.
[23] L. Badger, "Self-protecting mobile agents obfuscation techniques evaluation report," Netw. Associates Lab., Rockville, MD, USA, Tech. Rep. 01-036, 2002.
[24] J. Riordan and B. Schneier, "Environmental key generation towards clueless agents," in *Mobile Agents and Security*. Berlin, Germany: Springer, 1998, pp. 15–24.
[25] F. Linna and L. Jun, "A free-roaming mobile agent security protocol against colluded truncation attack without trusted third party," in *Proc. Int. Conf. Bus. Manage. Electron. Inf.*, vol. 2, May 2011, pp. 14–18.
[26] S. Srivastava and G. C. Nandi, "Fragmentation based encryption approach for self protected mobile agent," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 26, no. 1, pp. 131–142, Jan. 2014.
[27] M. A. Shaik, "Protecting agents from malicious hosts using trusted platform modules (TPM)," in *Proc. 2nd Int. Conf. Inventive Commun. Comput. Technol. (ICICCT)*, Apr. 2018, pp. 559–564.
[28] P. Bagga and R. Hans, "Mobile agents system security: A systematic survey," *ACM Comput. Surv.*, vol. 50, no. 5, p. 65, Nov. 2017.
[29] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, Jul./Oct. 1948.

BANDAR ALLUHAYBI received the B.S. degree in computer science from King Abdulaziz University, Saudi Arabia, in 2009, and the M.S. degree in engineering system management from St. Mary's University, San Antonio, TX, USA, in 2012. He is currently pursuing the Ph.D. degree in computer science with King Abdulaziz University. He is a Lecturer with the Computer Science Department, University of Prince Mugrin. His current research interests include information security, computer networks, networks security, big data, and high-performance computing.

MOHAMAD SHADY ALRAHHAL received the degree in computer engineering from Al-Baath University, Homs, Syria, in 2011, the master's degree from Damascus University, Syria, in 2013, and the Ph.D. degree from the Department of Computer Science, College of Computing and Information Technology, King Abdulaziz University, Saudi Arabia, in 2018. He was the Director of the IT Department, IoT Company, and the Risk Management Manager. His current research interests include agent-based software technology, artificial intelligent, cyber security, image and video processing, and risk management.

AHMED ALZAHRANI (Member, IEEE) received the B.S. degree in computer science from King Abdulaziz University, in 2000, the M.S. degree in information security from the University of Glamorgan, Cardiff, U.K., in 2005, and the Ph.D. degree in computer networks from the University of Bradford, U.K., in 2009. He is currently an Associate Professor with the Computer Science Department and the Vice Dean of Deanship of graduate studies for academic affairs with King Abdulaziz University. His current research interests include computer networks, networks security, quality of service routing, quantum computing, deep learning, big data, in-memory computing, and high-performance computing.

VIJEY THAYANANTHAN received the Ph.D. degree in engineering (digital communication engineering) from the Department of Communication Systems, University of Lancaster, U.K., in 1998. He is currently a Professor with Computer Science Department, King Abdulaziz University, Jeddah, Saudi Arabia. Since 2000, he has been working as a Research Engineer and a Senior Algorithm Development Engineer at Advantech Ltd., University of Southampton Science Park, U.K., and Amfax Ltd., U.K. His research interests include wireless communication algorithm design and mobile communication analysis, security management of communication networks and big data, computer security and wireless sensor networks. He has been a full-time member of the Institution of Engineering and Technology (IET), U.K., since 2005.