

A Lightweight Algorithm against Replica Node Attack in Mobile Wireless Sensor Networks using Learning Agents

Mojtaba Jamshidi¹, Shokooh Sheikh Abooli Poor², Nooruldeen Nasih Qader³,
Mehdi Esnaashari⁴, and Mohammad Reza Meybodi⁵

¹ Department of Information Technology, University of Human Development, Sulaymaniyah, Iraq
jamshidi.mojtaba@gmail.com

² Computer Engineering Department, Iranian Academic Center for Education, Culture and research (ACECR), Ahvaz, Iran
shokooh_sheikh@yahoo.com

³ Computer Science Department, University of Human Development, Sulaymaniyah, Iraq nooruldeen.qader@uhd.edu.iq

⁴ Faculty of Computer Engineering, K. N. Toosi University of Technology, Tehran, Iran esnaashari@kntu.ac.ir

⁵ Computer Engineering and Information Technology Department, Amirkabir University of Technology, Tehran, Iran
mmeybodi@aut.ac.ir

* Corresponding Author: Mojtaba Jamshidi

Received October 18, 2018; Revised February 2, 2019; Accepted February 7, 2019; Published February 28, 2019

* Regular Paper

Abstract: The node replication attack is considered one of the most dangerous attacks against wireless sensor networks (WSNs). In this attack, an adversary captures one or more normal nodes of the network, extracts its key materials, generates several replicas, and deploys them in the network. In this paper, we propose a novel, intelligent, and lightweight algorithm using learning agents and watchdog nodes (WNs) to detect replica nodes in mobile WSNs. In the proposed algorithm, there are a few WNs, each one equipped with a learning agent that monitors the network traffic and nodes' movements to identify potential replica nodes in the network. We use the fact that the number of times that a replica node meets a WN is probably more than that of a normal node during a specific monitoring interval of the network. The proposed algorithm is simulated using the J-SIM simulator, and its performance is evaluated in terms of replica node-detection probability and false detection probability through experiments. Experiment results show that the proposed algorithm can detect 100% of the replica nodes, while its false detection probability is less than 0.01.

Keywords: Mobile sensor networks, Learning agents, Replica nodes, Watchdog nodes

1. Introduction

A wireless sensor network (WSN) is comprised of a set of sensor nodes (SNs) that provide environment surveillance. These networks are widely employed in military, industry, health, and other sciences, and they are most appropriate for environments that are dangerous, difficult, or costly for a human presence. In each mission, many nodes are distributed in the network, and after deployment of the nodes or the end of the mission, the nodes cannot be collected or used again; therefore, the cost of each SN should be low. Considering the small size and low cost of nodes, there are many constraints on the SNs in terms of memory capacity, computational power, radio range, and energy. Considering these constraints and unattended deployment of the SNs, the wireless nature of

communications, and the ever-increasing application of these networks in military fields, establishing security in WSNs is critical and challenging; therefore, it has attracted many researchers [1, 2].

One of the serious attacks in WSNs is the node replication attack, or replica nodes. Because of the unattended deployment of nodes in the network, an adversary can capture one (or several) normal nodes and extract valuable information, including key materials, and create replica nodes using these key materials. Replica nodes contain specifications and data (including IDs and key materials) of the captured nodes. Thus, they can establish a shared key with other normal nodes of the network. Then, the adversary can distribute the replica nodes and launch various other attacks on the network. On one hand, the adversary controls these replica nodes, and

on the other hand, they have IDs and key materials that allow them to seem like certified (normal) nodes. Therefore, protocols that are used for secure communications allow the replica nodes to establish shared keys with other nodes and the base station. An adversary can exploit this situation in various ways. For instance, the adversary can easily observe a significant part of the network's traffic, which passes through the replica nodes, or destroy the mission of the sensors by injecting distorted data and disrupting common protocols of the sensor network, including clustering and data aggregation [3, 4].

Several algorithms [5-15] have been proposed to combat the replica node attack in static WSNs. However, these algorithms cannot be employed in mobile WSNs, since most of them rely on node positioning by a global positioning system or by transmitting location claims to witness nodes or specific places in the network, or they are particularly designed for specific topologies, like a grid. In mobile WSNs, these mechanisms (because of the continuous movement of the nodes) cannot be applied [16][17]. Other researchers [18-27] have proposed algorithms against replica nodes in mobile WSNs, but they have drawbacks, including high communications and memory overhead, not being scalable, having a complicated process, a low rate in detecting replica nodes, a requirement to determine the location of the nodes, or the use of public keys and digital signatures. In the next section, the main ideas of these algorithms are described.

In our previous work [28], we used watchdog nodes and a distributed bitwise-labeling mechanism based on nodes' movement behaviors to detect Sybil nodes in mobile WSNs. In this paper, a novel, intelligent, and lightweight algorithm based on learning agents and watchdog nodes is proposed to detect replica nodes in mobile WSNs, such that it eliminates the drawbacks of the existing algorithms. The proposed algorithm does not require determining nodes' locations or disseminating locations, and it does not require public keys (a digital signature) or complicated processes for detecting replica nodes. In the proposed algorithm, there are a few watchdog nodes which run the algorithm to detect replica nodes. The proposed algorithm includes three phases:

I. Configuring the watchdog nodes. Each watchdog node equipped by a learning agent which has an action vector and a probability vector.

II. Monitoring the network traffic and nodes' mobility.

In this phase, which is repeated ψ times, each watchdog nodes updates its learning agent when meeting other nodes.

III. Detecting replica nodes. In this phase, each watchdog node marks replica nodes according to its learning agent's status.

The rest of this paper is organized as follows. In Section 2, previous studies are reviewed. Section 3 describes learning agents, and Section 4 explains the system and its assumptions and the attack model. Section 5 describes the proposed algorithm. Section 6 explains the performance evaluation and simulation results. The last section concludes the paper.

2. Related Work

In this section, we investigate the existing algorithms that combat the replica node attack in static and mobile WSNs.

2.1 Algorithms in Static WSNs

Line-Selected Multicast (LSM) [5] is a well-known distributed probability algorithm, which uses a public encryption key. In LSM, neighboring nodes guide each node to issue location claims for witness nodes, which are selected randomly. Each witness node or intermediate node that receives more than one location claim for a particular node, such as u , detects u as a replica node. Randomized, Efficient, and Distributed (RED) [6] is a centralized algorithm where the main idea is to transmit location claims (with digital signatures) to locations of the network that are selected based on a random number periodically issued by a central point. Random Walk (RAWL) and Table-assisted Random Walk (TRAWL) [7] were proposed based on a random walk to detect replica nodes. In RAWL, for each node u , several random steps are taken, and nodes that pass are selected as witness nodes of u . The TRAWL algorithm is based on RAWL, but it adds a trace table to each node to reduce memory costs.

Jamshidi et al. [8] proposed a novel algorithm based on a dynamic ID-assignment mechanism to defend against a replica node attack in static WSNs. This algorithm uses a multi-tree architecture based on a multi-sink architecture for dynamic assignment of IDs to sensor nodes after deployment in a network environment. If this mechanism is used, replica nodes generated by the adversary cannot be easily attached to the network.

A message verification and passing method was applied by Maheswari and Kumar [8] to detect a cloned node. Yu et al. [10] proposed a clone detection mechanism based on a signal processing technology: compressed sensing. This mechanism bases its detection effectiveness on the compressed aggregation of sensor readings. Guan and Chen [11] proposed a watermarking mechanism based on clustering and sampling time intervals to detect clone attacks. A watermark is computed from one data sampling time. Then, it is embedded into fixed-size group data. The cluster completely reconstructs the watermark, and a statistical judge extracts watermark characteristics.

Mishra et al. [12] used the deviation in the distance traveled by a node, and its replica is discovered by observer nodes. Observer nodes keep a sliding window of recent time-distance broadcasts of the other nodes. A replica is detected by an observer node based on the degree of violation computed from the deviations recorded using the time-distance sliding window. To detect replica nodes in WSNs, Khan et al. [13] proposed a distributed algorithm that mingles the division of the network into areas with a random walk. In the first phase of this algorithm, the entire network is divided into different areas by using a heuristic-based algorithm. In the second phase, a replica node is detected by following a claimer-reporter-witness framework and a random walk.

Ding et al. [14] used similarity estimation with group

deployment knowledge to detect cloned nodes in WSNs. They prevent replicas from generating false location claims without deploying localization techniques on the sensor nodes. Roy and Nene [15] proposed an algorithm based on the received signal strength indicator (RSSI), link quality indicator (LQI) and packet sequence number (PSN) to detect replica nodes in WSNs. Also, they used a combination of broadcast and unicast messages. RSSI is used to obtain the amount of residual battery life, LQI is dependent on the distance between the transmitter and the receiver nodes, and the PSN is used to keep a check on any duplicate packet generated.

2.2 Algorithms in Mobile WSNs

The main idea of the algorithm proposed in [18] is that if SN u meets another node, v , at time T_1 , it sends a random number, r , to v at the same time. When nodes u and v meet each other once again at time T_2 , u asks v for the random number sent to it at time T_1 , and expects v to send it the same number, r . If v is not a replica node, it returns r , but if it is a replica node, it might return another random number. Disadvantages of this algorithm are a slow detection process and high overhead in communications and memory.

The main idea of the algorithm proposed in [19] was inspired by the fact that a legal node should not move faster than the maximum speed of the configured system. Replica nodes of u make other nodes think that node u moves faster than a predefined speed. In this case, they can mark u as a replica node. Disadvantages of these algorithms are high communications costs (each node should broadcast its identity and its neighbors' identity periodically), predefined maximum and minimum speed of nodes, the need for each node to be aware of its location, and requiring a public key and a digital signature.

The main idea of the algorithm in [20] is to use a pre-distributed, pair-wise key protocol based on polynomials and Bloom filters to guarantee that replica nodes cannot be located close to nodes with the real identities, and to collect some pair-wise keys established by each node. Disadvantages of this algorithm are its being centralized, not being scalable, its long detection process, and its marking of replica nodes.

Time domain detection (TDD) and spatial domain detection (SDD) [21] were proposed to detect replica node attacks in mobile networks. Moreover, the algorithm in [22] was proposed for mobile WSNs, and employs sign-based identity authentication to detect replica nodes.

Zhou and Wang [23] proposed a scheme to defend against the replication attack in mobile WSNs. This scheme has two levels: local detection and global detection. Local detection is performed in a local area much smaller than the whole deployed area to increase the probability of finding contradictory locations; global detection over a longer time period assigns an epoch verity location claim with every node it meets. This scheme is also described in [24] in more detail and with extensive experiments.

Dimitriou et al. [25] developed two distributed and decentralized schemes to defend against the replication attack in mobile WSNs. The main idea of these schemes

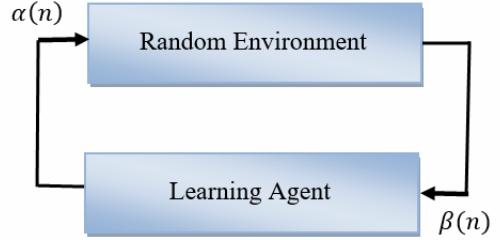


Fig. 1. The interaction between a learning agent and its environment.

was taken from the following consideration: two sensor nodes generate random nuances and send them to each other at their first meeting to be used as the values they exchanged in a previous meeting. At future meetings, if a node cannot reply correctly, or replies with the wrong nuance, it is treated as a malicious node, and the ID of that node is considered compromised.

Conti et al. [26] proposed two protocols to defend against replication attacks, called History Information-exchange Protocol (HIP) and its optimized version, HOP. Both of these protocols employ local information and node mobility to detect replica nodes. The two protocols differ in the amount of computation required.

Chowdhury and Dhawan [27] proposed a distributed replica detection algorithm that uses prefix matching. The main idea of this algorithm is to implement a decentralized key-based verification protocol to identify replica nodes. This algorithm was designed to work in mobile WSNs, and is based on using a random number and a hash function of the claiming node's ID in order to generate the keys.

3. Learning Agents

In this section, we propose a model for learning agents. The proposed learning agent model is inspired by learning automata [29, 30]. However, the proposed model differs from learning automata in such a way that right actions are rewarded. A learning agent is a machine with finite states, which can perform a limited number of actions. Each selected action is evaluated by a random environment, and the learning agent responds. The learning agent uses this response and selects its action for the next step. During this process, the agent learns to choose the best action from among the allowed actions. Fig. 1 shows the relation between the learning agent and the environment.

The environment can be represented with triplex $E = \{\alpha, \beta, c\}$ in which $\alpha = \{\alpha_1, \alpha_2, \dots, \alpha_r\}$ is the set of the environment's inputs, $\beta = \{\beta_1, \beta_2, \dots, \beta_m\}$ is the set of outputs, and $c = \{c_1, c_2, \dots, c_r\}$ is the set of penalty probabilities. The environment's input is one of r actions selected by the learning agent. The output (response) to each operation, i , is determined by β_i . The learning agent can be represented with a foursome, $LA = \{\alpha, \beta, p, T\}$, where $\alpha = \{\alpha_1, \alpha_2, \dots, \alpha_r\}$ is the set of agent actions (r is the number of actions), $\beta = \{\beta_1, \beta_2, \dots, \beta_r\}$ is the set of learning agent inputs, $p = \{p_1, p_2, \dots, p_r\}$ is the

probability vector of the learning agents' actions, and $T \equiv p(k+1) = T[\alpha(k), \beta(k), p(k)]$ is the learning algorithm. If the learning agent selects action α_i in time k , the probabilities of the actions are updated as follows: a is the reward parameter, and b is the penalty parameter [30, 31]:

Favorable response from the environment:

$$\begin{aligned} p_i(k+1) &= p_i(k) + a[1 - p_i(k)] \\ p_j(k+1) &= (1-a)p_j(k) \quad \forall j, j \neq i \end{aligned} \quad (1)$$

In this case, the learning agent selects just action α_i for time $(k+1)$.

Unfavorable response from the environment:

$$\begin{aligned} p_i(k+1) &= (1-b)p_i(k) \\ p_j(k+1) &= \frac{b}{r-1} + (1-b)p_j(k) \quad \forall j, j \neq i \end{aligned} \quad (2)$$

In this case, learning agent selects an action randomly and based on a probability vector for time $(k+1)$. Regarding the values of a and b in the above equations, three states can be considered as a learning automaton model. If $a=b$, the learning agent is called L_{RP} . When $b=0$, the learning agent is called L_{RI} , and if $b \ll a$, the learning agent is called L_{ReP} .

4. System Assumptions, the Attack Model, and Symbols

A sensor network contains two sets of nodes—sensor nodes (SNs) and watchdog nodes (WNs)—which are randomly distributed in a two-dimensional environment. The number of SNs is $m = |SNs|$, and the number of WNs is $\varpi = |WN|$. The number of WNs is much less than the number of SNs; n represents the total number of nodes, and n is obtained in the proposed algorithm by $n = \varpi + m$. SNs take on the network mission (like collecting information, sending data towards the base station, etc.) and WNs are responsible for detecting replica nodes. Each node has a unique ID and is unaware of its position. The radio range of all nodes is equal. All nodes are mobile and move according to mobility models, like two-dimensional mobility model (IID) [18, 26], throughout the network's lifetime. Nodes communicate with each other through a wireless radio channel and employ omnidirectional dissemination. SNs are not tamper-resistant. If an adversary captures a node, it can access its secret information and reprogram it. But we assume that WNs are tamper-resistant, and if an adversary captures them, the WNs cannot be decoded and reprogrammed [31, 32]. Additionally, since SNs are mobile, nodes should periodically (after each t time unit, or when they reach a new location in the network) broadcast a Hello message, a route request, send data, or send a keep-alive message [1, 28]. This broadcasting is one of the requirements of a

mobile WSN. Therefore, each node can detect its current neighbors, and if required, it can establish security keys with them, communicate with them, and create a routing table. WNs prevent sending such messages periodically so they remain hidden, because they are responsible for detecting malicious nodes (replica nodes).

Also, we assume that the sensor network is developed in an adversarial environment; thus, in such an unsecure network, an adversary can capture nodes, create copies of them, and inject them into the network. We also assume that each replica node (like normal nodes) in each period t broadcasts a Hello message, a routing request, transmits data, or sends a keep-alive message. Adversary nodes might set replica nodes in specific locations, or they could be mobile, just like normal nodes. None of these states affects the proposed algorithm.

In this paper, the following symbols are used:

- ϖ is the number of WNs in the network.
- m is the number of SNs in the network.
- n is the total number of nodes in the network.
- d is the number of neighbors on average
- M is the number of captured nodes by the adversary.
- R is the number of copies generated from each captured node.
- Ψ represents the number of times in which the second phase of the proposed algorithm should be executed.
- A_{vector} represents the action vector for each learning agent.
- P_{vector} represents the probability vector for each learning agent.
- a is the reward parameter for learning agents.
- b is the penalty parameter for learning agents.
- μ is the mean of probabilities in a P_{vector} ($\mu = \frac{1}{m}$).

5. The Proposed Algorithm

The main idea of the proposed algorithm is to employ the proposed learning agent and to have nodes broadcast standard messages (routing requests, data transmissions, or Hello and keep-alive messages) to detect replica nodes in mobile WSNs. As mentioned, in the proposed algorithm, in addition to SNs, there are a few WNs that monitor network traffic and detect replica nodes. A learning agent is mounted on each WN. The proposed learning agent model is effective due to its being intelligent, lightweight, and non-deterministic, which is suitable for mobile WSNs.

The proposed algorithm includes three phases. In the first phase, learning agents are configured. In the second phase, each WN updates its learning agent by monitoring the network traffic. This phase is broken into Ψ smaller periods with length t . In each period t , nodes select a random destination and become static for a time, and then start transmitting Hello packets, routing requests, etc. After Ψ monitoring rounds, the second phase terminated and the third phase starts, in which replica node detection starts.

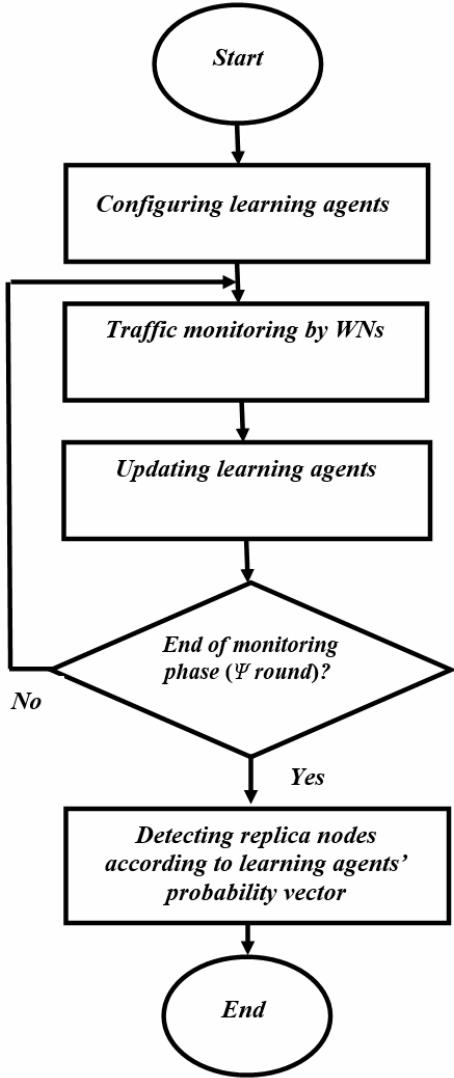


Fig. 2. Flowchart of the proposed algorithm.

A flowchart of the proposed algorithm is given in Fig. 2. In the following, details of these three phases are described.

5.1 Configuring the learning agents

Before nodes are deployed in the network environment, learning agents are loaded on WNs and, assuming that the number of SNs is η , the action vector (A_{vector}) and the probability vector (P_{vector}) of the learning agents are adjusted, as expressed in Eq. (3):

$$A_{vector} = [1.2 \dots m] \quad (3)$$

$$P_{vector} = \left[\frac{1}{m}, \frac{1}{m}, \dots, \frac{1}{m} \right]$$

In fact, each SN represents an action for the learning agent. Then, each learning agent selects an action (α_i) randomly. Action α_i selected by the learning agent in WN v , is in fact a node that v expects to meet in its neighborhood in the next monitoring round (the first

period of the second phase). After this step (that is, configuring learning agents on watchdogs), nodes are randomly deployed in the network.

5.2 Traffic Monitoring

Each node broadcasts a Hello message after becoming stationary to introduce itself to its neighbors, and if required, it sends a routing request, some data, or a keep-alive message. This broadcast lets the WN know which nodes are currently in its neighborhood. After t time units, when the current period is terminated, the learning agent of each watchdog v receives a response (positive or negative) from the environment. If selected action (or node) α_i is in the neighborhood of WN v , it means that the response is positive; otherwise, the response is negative. If a learning agent receives a positive response from the environment, it rewards action α_i according to Eq. (1), and selects this α_i for the next monitoring round. But if it receives a negative response from the environment, it first penalizes action α_i according to Eq. (2), and then randomly selects another action based on P_{vector} for the next monitoring round. This is done by the WNs simultaneously. So, when period t passes, the first time period of the second phase of the proposed algorithm terminates. Then, nodes select a new random destination for themselves and move towards it; the next round of the second phase starts.

As mentioned, the second phase executes ψ rounds. Since, in a replica node attack, an adversary captures a node (for example, u) and disseminates several copies of it in the network, a node with ID u appears in the neighborhood of the WNs more than usual, and this increases the action probability corresponding to this replica node in the probability vector, so it tends to 1.

5.3 Detecting Replica Nodes

When the second phase of the proposed algorithm terminates, each WN should decide whether it has detected a replica node or not. If there is no replica node in the network, then according to a random model of nodes' movements, the number of times that a node has appeared in the neighborhood of a WN would be almost equal for all nodes (that is, the probabilities of the P -vectors is equal). But if an adversary captures a node, u , and injects several copies of it into the network, then the number of times that node u appears in the neighborhood of WNs would be more than it is for the other nodes. This difference makes the probability of the action corresponding to node u in P -vector of the WNs to be more than the probability corresponding to other nodes' actions. Therefore, each WN should detect replica nodes according to the probability vector (P_{vector}) of their learning agent. Here, a simple procedure is proposed to detect replica nodes based on P -vector. The pseudo-code of this procedure is in Fig. 3.

In this procedure, the mean (μ) of the P -vector is calculated first. Then, this P -vector is sorted in descending order. Finally, a loop is used to scan P -vector, and if the difference of the i th and the $(i+1)$ th action in terms of their probability values is more than two times P_{vector} 's mean

```

1: evaluate  $\mu$  ,  $\mu = \frac{1}{m}$ 
2: sort  $P_{vector}$  in descend order
3: malicious_list =  $\varnothing$ 
4: for  $i = 0$  to  $m - 2$ 
    if( $P_{vector}[i] - P_{vector}[i+1] > 2\mu$ )
        add  $A_{vector}[i]$  to malicious_list
    else
        exit for

```

Fig. 3. Pseudo-code of replica node detection procedure.

(μ), the action corresponding to $P_{vector}[i]$, means $A_{vector}[i]$ is marked as a replica node. But if the conditions are not satisfied for the i th element, the scanning loop is broken, and elements greater than the i th element are not scanned because the probability of actions corresponding to replica nodes in the probability vector is more than other actions. Thus, the difference in the probabilities of actions corresponding to replica nodes would be more than two times the mean, compared to an action corresponding to non-replica nodes (according to the number of rounds in the second phase and the number of copies).

Then, the WNs give the list of detected replica nodes to the base station. Thus, the base station informs all nodes of the network, or it directly broadcasts the list to the network, so that legal nodes do not communicate with these malicious nodes anymore.

Note: The proposed algorithm might fail in detection. That is, some non-replica nodes might be detected as replica nodes. False detection occurs when some normal nodes (like v) appear a large number of times in the neighborhood of a specific WN (like w), which increases the probability corresponding to action v in the P_{vector} of WN w . Therefore, in the third phase of the proposed algorithm, node v is detected as a replica node by the WN. The total number of nodes in the network (n), the number of traffic monitoring rounds (ψ), the number of nodes captured by an adversary (M), and the number of WNs (ϖ) are the parameters that affect the false detection rate. Experiment results in the following section demonstrate this problem.

6. Performance Evaluation and Simulation Results

In this section, we evaluate the performance of the proposed algorithm with regard to memory, communications, and computation overhead. We compare the results with other algorithms. Then, we demonstrate simulation results of the proposed algorithm.

6.1 Performance Evaluation

Memory Overhead: Since the proposed algorithm is

Table 1. Comparing memory and communications overhead of the proposed algorithm with other algorithms.

Algorithm	Memory overhead (per node)	Communication overhead (per network)
LSM [5]	$O(\sqrt{n})$	$O(n\sqrt{n})$
RED [6]	$O(d)$	$O(n\sqrt{n})$
RAWL [7]	$O(\log n \times \sqrt{n})$	$O(\log n \times \sqrt{n})$
XED [18]	$O(4 \times d \times E[X])$	$O(n \times d \times \psi)$
SPRT [19]	$O(n\sqrt{n})$	$O(n)$
Algorithm in [20]	$O(d)$	$O(n \times \log n)$
TDD, SDD [21]	$O(n)$	$O(\sqrt{n}), O(d)$
The proposed algorithm	$0 \sim O(n) \dagger$	0

n : number of nodes

d : number of neighbors on average

w : number of witness nodes that store location claims of L

s : number of SNs in one cell of the network

$E[X]$: number of expected movements that a node needs to meet two different copies of replica nodes

\dagger for SNs it is zero, and for WNs it is $O(n)$

Hint: In the proposed algorithm, overhead (memory and communications) is imposed on WNs only, but in other algorithms, overhead is imposed on all nodes.

executed only by WNs, it does not impose any memory overhead on SNs. But each WN requires $2m$ in memory to storing its A_{vector} and P_{vector} . Since m is close to n , we can suppose that the memory overhead of each WN is on the order of $O(n)$. Table 1 compares the memory overhead of the proposed algorithm with other existing algorithms. As we can see in Table 1, the proposed algorithm performs better than other algorithms in terms of memory overhead, because its memory overhead is 0 for m SNs, and it is on the order of $O(n)$ for ϖ WNs, where it should be noted that the number of SNs is much higher than the number of WNs ($m \gg \varpi$).

Communications Overhead: Considering the energy limitations of the SNs, the energy consumed by the proposed algorithm for sensor networks is crucial. Sending, receiving, and processing are the critical operations that consume energy. Since sending packets consumes more energy than processing and receiving packets, calculating the number of sent packets, which is imposed by the network due to using a specific algorithm, is considered the important measure for evaluating the efficiency of the algorithms. Since the proposed algorithm only uses the Hello message, routing requests, data transmissions, and keep-alive messages to execute, and since transmitting these messages is the default in a mobile sensor network, no communications overhead is imposed on the SNs. Also, WNs do not send a message to the network during various rounds of the proposed algorithm. Thus, the communications overhead for WNs is also zero. Table 1

compares communications overhead of the proposed algorithm and other algorithms. The proposed algorithm outperforms other algorithms in terms of communications overhead.

Computation Overhead: Since the proposed algorithm is executed by WNs, no computation overhead is forced on SNs. But each WN should update its learning agent after each round of the second phase. Besides, each WN in the third phase needs $O(n \log n)$ in time to sort its P_{vector} .

6.2 Simulation Model

The proposed algorithm was simulated by using the simulator J-SIM [33], and its performance was evaluated regarding detection probability and false detection probability metrics through experiments.

Detection probability (P_s): this metric is calculated by the following [7]:

$$P_s = \frac{\# \text{successful detection times}}{\# \text{repeat times}} \quad (3)$$

Indeed, it should be noted that each execution of the proposed algorithm and other algorithms, like XED, RED, RAWL, and LSM, include ψ rounds.

False detection probability (P_f): the probability that a non-replica node is incorrectly detected as a replica node by a security detection algorithm.

In the simulations, it is assumed that the network consists of n SNs that are distributed randomly in a 100×100 meters area. An adversary captures M legal nodes and generates R copies from each. In other words, the network environment contains $R \times M$ malicious nodes. Additionally, the number of WNs is considered to be ϖ . Reward and penalty parameters of the learning agents were set at $a=0.01$ and $b=0.001$ (except for the last experiment). The radio range of all nodes was considered to be 20 meters. We used the two-dimensional IID mobility model [1, 18, 28, 29, 34] to verify the results. In the IID mobility model, at beginning of each round, sensors are uniformly and randomly placed in the network, so that the position of each node is independent from the one of the previous round [18, 28]. Each simulation was repeated 50 times, and then, the results were averaged to obtain a final value.

6.3 Experiment Results

Experiment 1: In this experiment, we set the parameters at $\varpi=10$, $M=5$, $R=10$, and $n=100$. Detection probability and false detection probability were evaluated for $\psi=50, \dots, 400$. Figs. 4 and 5 show the experiment results for detection probability and false detection probability, respectively. Fig. 4 shows that the probability of detecting replica nodes for $\psi=200$ is 0.84; for $\psi=300$, it is 0.98, and for $\psi=350$, it is 1.

The results in Fig. 5 show that the false detection probability of the proposed algorithm for $\psi < 350$ is less than 0.005, and for $\psi \geq 350$, it is 0. It is obvious why

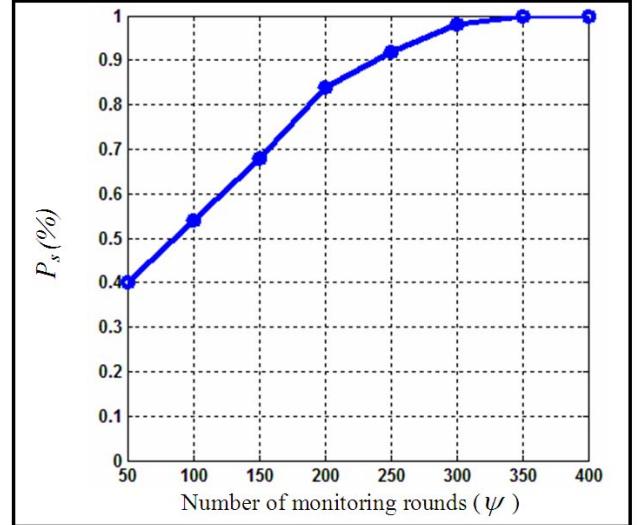


Fig. 4. Effect of the number of monitoring rounds, ψ , on detection.

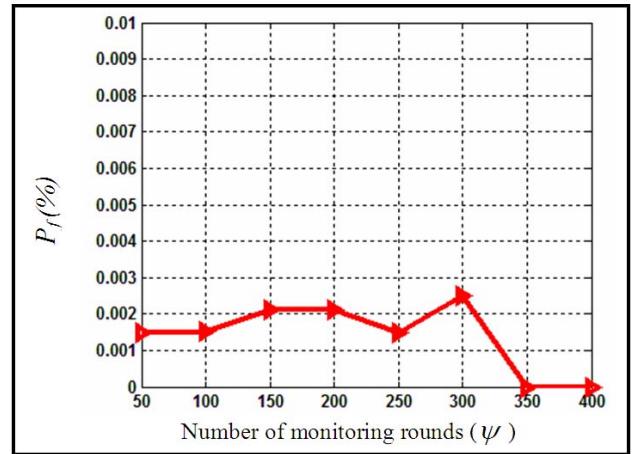


Fig. 5. Effect of the number of monitoring rounds, ψ , on the false detection probability.

these results were obtained; as the number of traffic monitoring rounds increases, WNs meet replica nodes in their neighborhood more often, which makes the probability of actions corresponding to replica nodes tend towards 1, and the probability of other actions (non-replica nodes) tends towards 0. Therefore, the third phase of the algorithm discriminates replica nodes from non-replica nodes with higher accuracy. Therefore, increasing parameter ψ directs the detection probability towards 1 and the false detection probability towards 0.

Also, the performance of the proposed algorithm was compared to other existing algorithms, LSM, RED, XED, HIP (h=5), and HOP (h=5), in terms of detection probability for $\psi=300$. As Fig. 6 illustrates, the detection probability in the proposed algorithm is higher than the other algorithms.

Experiment 2: The goal of this experiment was evaluating the effect of the number of nodes (n) on the performance of the proposed algorithm. In this experiment,

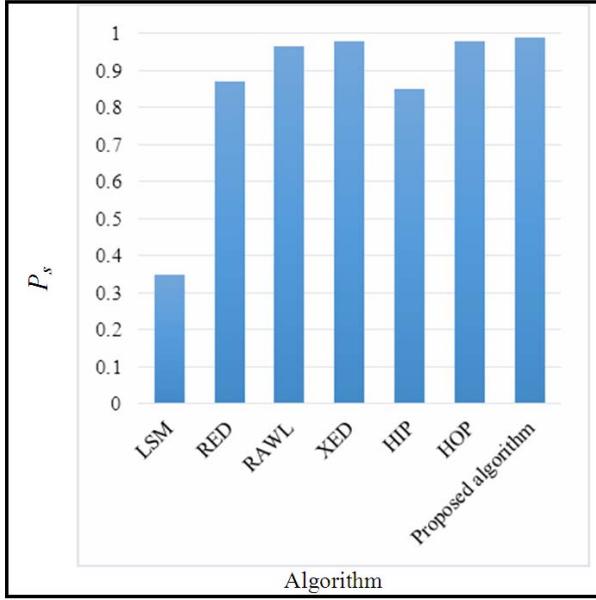


Fig. 6. Comparison of the performance of the proposed algorithm and other algorithms in terms of detection probability probability for $\psi > 300$.

we set the parameters at $\varpi = 10$, $M = 5$, $R = 10$, and $\psi = 350$. Detection probability and false detection probability were evaluated for $n=100$, 200, and 300, and the results are given in Fig. 7. Results of this experiment showed that increasing the number of nodes decreases the detection probability and increases the false detection probability. That is, as the number of nodes increases, the number of learning agents' actions also increases. On the other hand, as the number of learning agents' actions increases, the convergence speed of the learning agent decreases. For instance, when there are 100 nodes in the network, detection and false detection probabilities of the proposed algorithm are 1 and 0, respectively. However, when there are 300 nodes in the network, detection and false detection probabilities are 0.92 and 0.014, respectively. Indeed, if the number of rounds in the second phase, ψ , is increased ($\psi > 350$), the probability of detecting replica nodes exceeds 0.92 and tends towards 1, and the false detection probability becomes less than 0.014 and tends towards zero. In order to verify this statement, the experiment is repeated for $\psi > 500$, and the results are shown in Fig. 8. We can see that by increasing the number of traffic monitoring rounds, the performance of the algorithm improves.

Experiment 3: The goal of this experiment was evaluating the effect of the number of replica nodes (for any captured node) injected into the network (R) on the performance of the proposed algorithm. In this experiment, we set the parameters at $\varpi = 10$, $M = 5$, $n = 100$, and $\psi = 300$. We evaluated the detection probability and the false detection probability of the proposed algorithm for $R=5$, 10, and 15. The results of this experiment in Fig. 9 show that by increasing R , the probability of detecting replica nodes also increases. In the following, we analyze

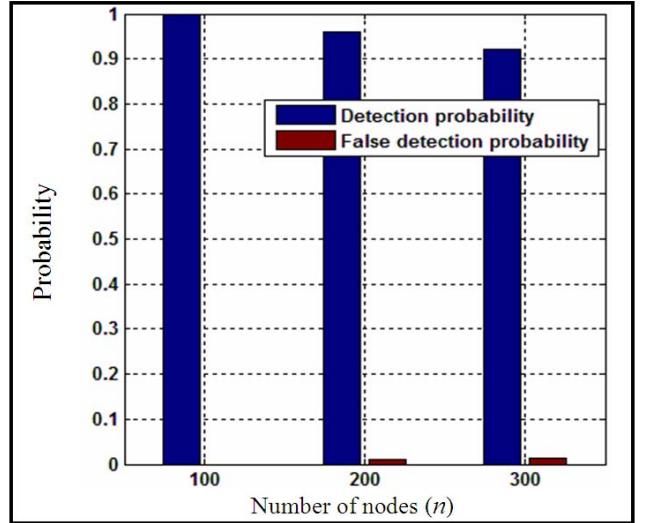


Fig. 7. Effect of the number of nodes, n , on the detection probability and false detection probability, for $\psi > 350$.

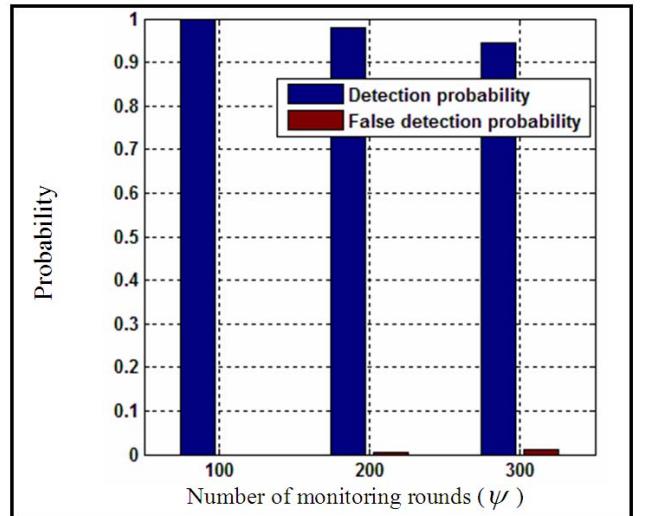


Fig. 8. Effect of the number of nodes, n , on the detection probability and false detection probability, for $\psi > 500$.

the result.

An adversary can disrupt the performance of the network by establishing replica nodes in two ways. First, the adversary captures a high number of legal nodes, generates a few copies of each of them, and deploys them in the network. In this case, security algorithms can hardly ever detect replica nodes. However, using this way might be challenging and time-consuming for the adversary because it must capture, decode, reprogram, and control a large number of nodes. Second, the adversary captures a few nodes and generates a high number of copies from each of them and deploys them in the network. The second way is easy and feasible for an adversary. Nevertheless, security algorithms might detect replica nodes faster and more accurately. The result of this experiment also showed that the proposed algorithm detects replica nodes faster for

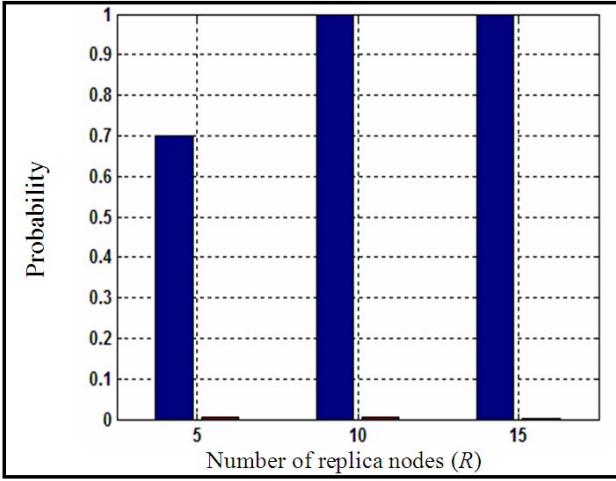


Fig. 9. Effect of parameter R on the detection probability and false detection probability for $\psi = 300$.

a larger R . When there is a high number of copies from a captured node, like u , WNs meet u more often. Therefore, the probability value corresponding to the action of this node in the learning agents increases faster and tends towards 1. Additionally, results showed that changing parameter R does not affect false detection probability, and it is between 0.003 to 0.004.

Experiment 4: The purpose of this experiment was evaluating the effect of the number of WNs (ϖ) on the performance of the proposed algorithm. We set the parameters of this experiment at $n=100$, $M=5$, $R=10$, and $\psi=600$. Detection probability and false detection probability for the proposed algorithm were evaluated for $\varpi=2, \dots, 10$. Fig. 10 shows the experiment results for detection probability, and Fig. 11 shows the experiment results for false detection probability. Experiment results show that increasing the number of WNs in the network increases the probability of detecting replica nodes. Because increasing the number of WNs brings more areas of the network under monitoring, that increases the probability of detecting all malicious nodes in the network. On the other hand, as mentioned in Section 5.3, it is probable that some SNs (e.g., u) appear repeatedly in the neighborhood of a particular WN, (e.g., w), which increases the probability corresponding to the action of node u in the P_{vector} of w . Therefore, in the third phase of the proposed algorithm, node u is incorrectly detected by WN w as a replica node. Increasing the number of WNs in the network increases the false detection probability, which increases the probability of a non-replica node appearing randomly and repeatedly in the neighborhood of a WN.

Experiment 5: The purpose of this experiment was investigating the effect of the number of captured nodes (M) on the performance of the proposed algorithm. We set the parameters in this experiment at $\varpi=10$, $n=100$, $R=10$, and $\psi=300$. We evaluated the detection probability and false detection probability for $M=0, 1, 3, 5, 7$. Fig. 12 shows the experiment results for detection probability, and Fig. 13 shows the results for

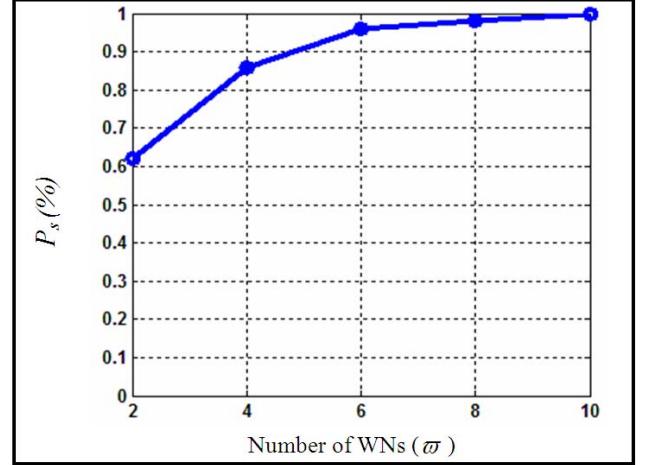


Fig. 10. Effect of the number of WNs, ϖ , on the detection probability.

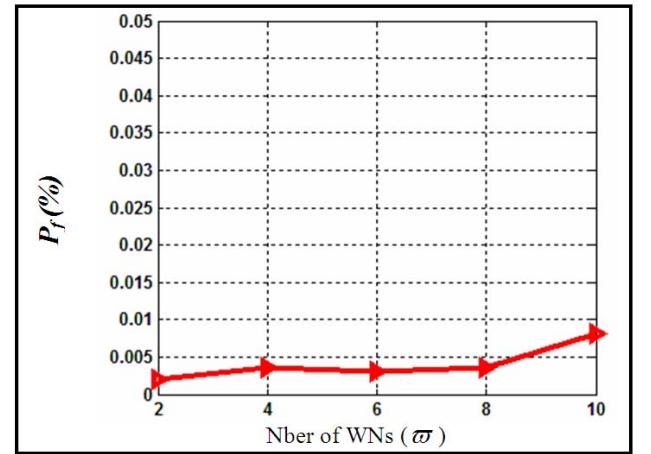


Fig. 11. Effect of the number of WNs, ϖ , on the false detection probability.

false detection probability.

Experiment results show that increasing the number of nodes captured by an adversary and increasing number of replica nodes (R) in the network, decreases the detection probability of replica nodes, for instance, if an adversary has captured fewer than four (i.e., $M=4$) nodes of the network and has generated $R=10$ copies from each node. The proposed algorithm detects 100% of the replica nodes after $\psi=300$. However if the number of nodes captured by the adversary reaches seven (i.e., $M=7$), the probability of the proposed algorithm detecting all these captured nodes after $\psi=300$ traffic monitoring rounds would be 0.90. Indeed, if the number of traffic monitoring rounds is increased, detection probability increases and reaches 1.

Also, Fig. 13 shows that increasing the number of nodes captured by the adversary decreases the false detection probability. By increasing the number of captured nodes and increasing the number of replica nodes in the network, a larger portion of P_{vector} is assigned to these captured nodes. Thus, the probability value for normal nodes decreases. Therefore, the probability that the

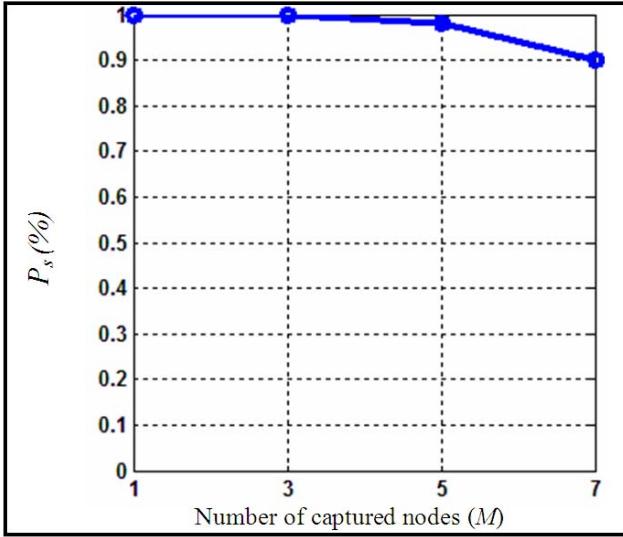


Fig. 12. Effect of the number of captured nodes on the detection probability.

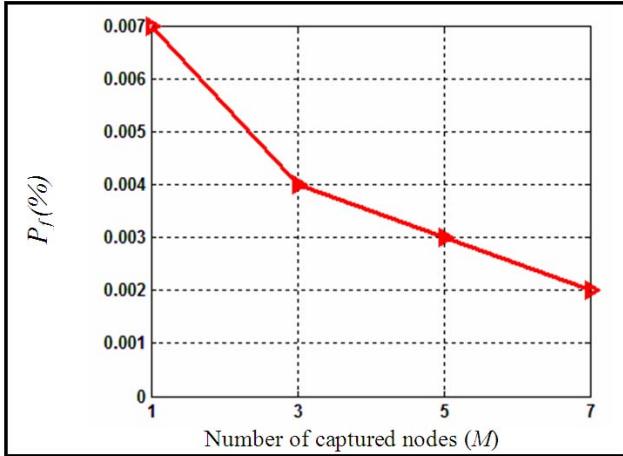


Fig. 13. Effect of the number of captured nodes on the false detection probability.

proposed algorithm detects normal nodes as replica nodes decreases. The result of this experiment showed that if the adversary only captured one node (i.e., $M=1$), false detection probability is 0.007, and if it has captured seven nodes (i.e., $M=7$), the probability would be 0.002.

Experiment 6: The purpose of this experiment was investigating the effect of different learning agent models, reward (a), and penalty (b) parameters, on the performance of the proposed algorithm. We set the parameters at $M = 7$, $\sigma = 10$, $n = 100$, $R = 10$, and $\psi = 300$. We evaluated the detection probability and false detection probability for different learning agent models and different values for the reward and penalty.

Experiment results in Table 2 showed that learning agent L_{ReP} gives the best result for $a=0.04$ and $b=0.001$, where the detection probability is 0.9741, and false detection probability is 0.005. In this type of learning agent, a small value for the reward parameter causes the learning agents to meet more slowly. Thus, they detect replica nodes more slowly. For instance, when $a=0.01$, the

Table 2. Effect of different learning agent models and reward and penalty parameters on the performance of the proposed algorithm.

Learning agent model	a	b	Detection probability	False detection probability
L_{ReP}	0.01	0.001	0.9	0.002
	0.02	0.001	0.985	0.005
	0.03	0.001	0.9714	0.007
	0.04	0.001	0.9717	0.005
	0.05	0.001	0.9571	0.004
	0.06	0.001	0.9285	0.007
	0.07	0.001	0.9	0.002
	0.08	0.001	0.8857	0.006
	0.1	0.001	0.8714	0.003
L_{RP}	0.001	0.001	0	0
	0.01	0.01	0.5571	0
	0.05	0.05	0.7142	0.029
	0.1	0.1	0.5857	0.045
L_{RI}	0.001	0	0	0
	0.01	0	0.9142	0
	0.05	0	0.914	0.002
	0.1	0	0.7857	0.009

detection probability would be 0.9. On the other hand, if the reward is significant (e.g., $a=0.1$), the detection probabilities decrease. The probability vector of the learning agent would be most affected only by the recent rewards (i.e., it is not affected by all rewards that have been awarded throughout the algorithm's execution). As mentioned before, the main idea of the proposed algorithm is to use the mobility of nodes in all traffic monitoring rounds. Thus, the proposed algorithm gives better results when the mobility of the nodes in all rounds for monitoring traffic has a similar impact on P_{vector} , which is true for other learning agent models also.

L_{RP} learning agents do not give desirable results because it assigns the same reward (if it receives the desired response from the environment) and the same penalty (if it receives an undesirable response from the environment) to the selected action. While the main idea of the proposed algorithm is mostly affected by the times that nodes appear in the neighborhood of the WN, it means it does not affect the times that nodes do not appear in the neighborhood. Results of this experiment for the L_{RI} learning agent model verify this issue. They show that when the penalty is zero (L_{RI} model), the model obtained better results compared to when the penalty is high (L_{RP} agent). However, results obtained from L_{ReP} are better than L_{RI} because in L_{RI} , when nodes are not present in the neighborhood of the WNs, the probability vector is not affected. Nodes in L_{ReP} that do not appear in the neighborhood of the WNs affect the probability vector a little, and this gives better results. In total, the experiment results show that the L_{ReP} learning agent model achieves better results compared to L_{RI} and L_{RP} learning agent models.

7. Conclusion

In this paper, we proposed a novel, intelligent, and lightweight algorithm using learning agents to detect replica nodes in mobile WSNs. In the proposed algorithm, learning agents and the local traffic of the nodes are used to detect replica nodes. We evaluated the overhead of communications, memory, and processing under the proposed algorithm. We simulated the proposed algorithm using the J-SIM simulator, and experiment results showed that the proposed algorithm could detect 100% of the replica nodes, whereas its false detection probability was less than 0.01. We compared the results with other existing algorithms. Comparison results showed that the proposed algorithm outperforms all other algorithms.

One of the disadvantages of the proposed algorithm is low convergence speed in the learning agents (particularly when there are many nodes in the network). In other words, the traffic monitoring phase of the proposed algorithm has to be repeated many times in order to achieve a high detection rate. But the following should be noted when executing these rounds of the traffic monitoring phase.

I. It does not impose any memory, communications, or computation overhead on the SNs.

II. It does not impose any memory or communications overhead on the WNs.

III. It does not impose any extra computation overhead on WNs in the detection phase (the third phase).

IV. Just a little computation overhead is imposed on WNs for updating P_{vector} , on the order of $O(n)$.

Hence, although the detection speed of the proposed algorithm is low, it does not impose significant overhead on the sensor nodes. Anyway, employing cellular learning automata can be an effective approach to increasing the convergence speed of the learning agents, which we will address in future study.

References

- [1] M. Jamshidi, M. Ranjbari, M. Esnaashari, N. N. Qader, M. R. Meybodi, “Sybil Node Detection in Mobile Wireless Sensor Networks Using Observer Nodes,” JOIV: International Journal on Informatics Visualization, 2(3): 159-165, 2018. [Article \(CrossRef Link\)](#)
- [2] M. Jamshidi, E. Zangeneh, M. Esnaashari, A. M. Darwesh, M. R. Meybodi, “A Novel Model of Sybil Attack in Cluster-Based Wireless Sensor Networks and Propose a Distributed Algorithm to Defend It,” Wireless Personal Communications, 1-29, 2019 (in press). [Article \(CrossRef Link\)](#)
- [3] A. K. Mishra, A. K. Turuk, “A comparative analysis of node replica detection schemes in wireless sensor networks,” Journal of Network and Computer Applications, 61: 21-32, 2016. [Article \(CrossRef Link\)](#)
- [4] H. R. Shaukat, F. Hashim, A. Sali, M. F. Abdul Rasid, “Node replication attacks in mobile wireless sensor network: A survey,” International Journal of Distributed Sensor Networks, 10(2): 1-15, 2014. [Article \(CrossRef Link\)](#)
- [5] B. Parno, A. Perrig, V. D. Gligor, “Distributed Detection of Node Replication Attacks in Sensor Networks,” IEEE Symposium on Security and Privacy: 49-63, 2005. [Article \(CrossRef Link\)](#)
- [6] M. Conti, R. D. Pietro, L. V. Mancini, A. Mei, “Distributed Detection of Clone Attacks in Wireless Sensor Networks,” IEEE transactions on dependable and secure computing, 8(5): 685-698, 2011. [Article \(CrossRef Link\)](#)
- [7] Y. Zeng, J. Cao, S. Zhang, S. Guo and L. Xie, “Random-Walk Based Approach to Detect Clone Attacks in Wireless Sensor Networks,” IEEE Journal on selected areas in communications, 28(5): 677-691, 2010. [Article \(CrossRef Link\)](#)
- [8] M. Jamshidi, A. A. Shalooki, Z. D. Zadeh and A. M. Darwesh, “A Dynamic ID Assignment Mechanism to Defend Against Node Replication Attack in Static Wireless Sensor Networks,” JOIV: International Journal on Informatics Visualization. 3(1): 13-17, 2019. [Article \(CrossRef Link\)](#)
- [9] P. U. Maheswari and P. G. Kumar, “Dynamic Detection and Prevention of Clone Attack in Wireless Sensor Networks,” Wireless Personal Communications, 94(4): 2043-2054, 2017. [Article \(CrossRef Link\)](#)
- [10] C. M. Yu, C. S. Lu and S. Y. Kuo, “Compressed Sensing-Based Clone Identification in Sensor Networks,” IEEE Transactions on Wireless Communications, 15(4): 3071-3084, 2016. [Article \(CrossRef Link\)](#)
- [11] T. Guan, Y. Chen, “A node clone attack detection scheme based on digital watermark in WSNs,” IEEE International Conference on Computer Communication and the Internet (ICCCI), 257-260, 2016. [Article \(CrossRef Link\)](#)
- [12] A. K. Mishra, A. K. Tripathy, A. Kumar, A. K. Turuk, “A Replica Detection Scheme Based on the Deviation in Distance Traveled Sliding Window for Wireless Sensor Networks,” Wireless Communications and Mobile Computing, 2017: 1-8, 2017. [Article \(CrossRef Link\)](#)
- [13] W. Z. Khan, M. Y. Aalsalem, N. M. Saad, Y. Xaing, T. H. Luan, “detecting replicated nodes in Wireless Sensor Networks using random walks and network division,” Wireless Communications and Networking Conference (WCNC), 2623-2628, 2014. [Article \(CrossRef Link\)](#)
- [14] C. Ding, L. Yang, M. Wu, “Localization-Free Detection of Replica Node Attacks in Wireless Sensor Networks Using Similarity Estimation with Group Deployment Knowledge,” Sensors, 17(1), 160-152, 2017. [Article \(CrossRef Link\)](#)
- [15] S. Roy, M. J. Nene, “Prevention of node replication in Wireless Sensor Network using Received Signal Strength Indicator, Link Quality Indicator and Packet Sequence Number,” IEEE International Conference on Green Engineering and Technologies (IC-GET), 1-8, 2016. [Article \(CrossRef Link\)](#)
- [16] S. Smys, J. S. Raj, “A self-organized structure for mobility management in wireless networks,” Computers & Electrical Engineering, 48: 153-163, 2015. [Article \(CrossRef Link\)](#)

- [17] Z. Pala, K. Bicakci, M. Turk, "Effects of node mobility on energy balancing in wireless networks," *Computers & Electrical Engineering*, 41: 314-324, 2015. [Article \(CrossRef Link\)](#)
- [18] C. M. Yu, C. S. Lu, S. Y. Kuo, "Mobile Sensor Network Resilient Against Node Replication Attacks," *IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, 597-599, 2008. [Article \(CrossRef Link\)](#)
- [19] J. W. Ho, M. Wright, S. Das, "Fast Detection of Mobile Replica Node Attacks in Wireless Sensor Networks Using Sequential Hypothesis Testing," *IEEE transactions on mobile computing*, 10(6): 767-782, 2011. [Article \(CrossRef Link\)](#)
- [20] X. M. Deng, Y. Xiong, "A new protocol for the detection of node replication attacks in mobile wireless sensor networks," *Journal of Computer Science and Technology*, 26(4): 732-743, 2011. [Article \(CrossRef Link\)](#)
- [21] K. Xing, X. Cheng, "From Time Domain to Space Domain: Detecting Replica Attacks in Mobile Ad Hoc Networks," *IEEE INFOCOM*, 1-9, 2011. [Article \(CrossRef Link\)](#)
- [22] W. T. Zhu, J. Zhou, H. Robert, D. F. Bao, "Detecting node replication attacks in mobile sensor networks: theory and approaches," *Security and Communication Networks*, 5(5): 496-507, 2012. [Article \(CrossRef Link\)](#)
- [23] C. Zhou, Z. Wang, "An Two Dimension detection to node replication attacks in mobile sensor networks," *10th IEEE International Conference on Anti-counterfeiting, Security, and Identification (ASID)*, 63-69, 2016. [Article \(CrossRef Link\)](#)
- [24] Z. Wang, C. Zhou, Y. Liu, "Efficient Hybrid Detection of Node Replication Attacks in Mobile Sensor Networks," *Mobile Information Systems*, 2017: 1-13, 2017. [Article \(CrossRef Link\)](#)
- [25] T. Dimitriou, E. A. Alrashed, M. H. Karaata and A. Hamdan, "Imposter detection for replication attacks in mobile sensor networks," *Computer Networks*, vol. 108, pp. 210-222, Oct. 2016. [Article \(CrossRef Link\)](#)
- [26] M. Conti, R. D. Pietro, A. Spognardi, "Clone wars: Distributed detection of clone attacks in mobile WSNs," *Journal of Computer and System Sciences*, 80(3): 654-669, 2014. [Article \(CrossRef Link\)](#)
- [27] A. S. Chowdhury, A. Dhawan, "Distributed Clone Detection in Mobile Sensor Networks," In *SENSORNETS*, 135-141, 2012. [Article \(CrossRef Link\)](#)
- [28] M. Jamshidi, E. Zangeneh, M. Esnaashari, M. R. Meybodi, "A lightweight algorithm for detecting mobile Sybil nodes in mobile wireless sensor networks," *Computers & Electrical Engineering*, 64: 220-232, 2017. [Article \(CrossRef Link\)](#)
- [29] M. Ranjbari, J. A. Torkestani, "A learning automata-based algorithm for energy and SLA efficient consolidation of virtual machines in cloud data centers," *Journal of Parallel and Distributed Computing*, 113: 55-62, 2017. [Article \(CrossRef Link\)](#)
- [30] M. Esnaashari, M. R. Meybodi, "Dynamic irregular cellular learning automata," *Journal of Computational Science*, 24: 358-370, 2017. [Article \(CrossRef Link\)](#)
- [31] E. Shi, A. Perrig, "Designing secure sensor networks," *IEEE Wireless Communications*, 11(6): 38-43, 2004. [Article \(CrossRef Link\)](#)
- [32] C. Tumrongwittayapak, R. Varakulsiripunth, "Detecting Sinkhole Attacks in Wireless Sensor Networks," *ICROS-SICE*, Fukuoka International Congress Center, 1966-1971, 2009. [Article \(CrossRef Link\)](#)
- [33] A. Sobeih, J.C. Hou, L.C. Kung, *et al.*: J-Sim: a simulation and emulation environment for wireless sensor networks. *IEEE Wireless Communications*, 13(4): 104-119, 2006. [Article \(CrossRef Link\)](#)
- [34] M. Jamshidi, A. M. Darwesh, A. Lorenc, M. Ranjbari, M. R. Meybodi, "A Precise Algorithm for Detecting Malicious Sybil Nodes in Mobile Wireless Sensor Networks," *IEIE Transactions on Smart Processing & Computing*, 7(6): 457-466, 2018. [Article \(CrossRef Link\)](#)



Mojtaba Jamshidi received the B.S. degree in Computer Engineering from the Iranian Academic Center for Education, Culture and research (ACECR), Kermanshah, Iran, in 2009, and M.S. degree in Computer Engineering from Islamic Azad University, Qazvin, Iran, in 2012. His research interests include computer networks, learning systems, security, meta-heuristic algorithms, data mining, and recommender systems.



Shokooh Sheikh Abooli Poor received the B.S. degree in Computer Engineering from the Islamic Azad University, Ramhormoz, Iran, in 2013, and M.S. degree in Computer Engineering from Islamic Azad University, Broujerd, Iran, in 2017. Her research interests include wireless networks, learning systems, security, data mining, and recommender systems.



Nooruldeen N. Qader received Computer Science Ph.D from University of Sulaimani, Iraq. He is currently working as Dean of College of Science and Technology in University of Human Development, Sulaymaniyah, Kurdistan Region of Iraq. He is senior member of IEEE. He has got international grants of MARHABA Erasmus Mundus lot 3 project 2014-0653 and MENA Scholarship Program. MENA was set up by the Netherlands Ministry of Development Cooperation for E-Government at Maastricht School of Management. He has 22 years of teaching and 12 years of research experience. His research areas are Information Security, Database, Cloud computing, Big data, and Data Mining.



Mehdi Esnaashari received the B.S., M.S. and Ph.D. degrees in Computer Engineering all from the Amirkabir University of Technology in Iran, in 2002, 2005, and 2011 respectively. He worked at Iran Telecommunications Research Center as an Assistant Professor from 2012 to 2016.

Currently, he is an Assistant Professor in Computer Faculty of K. N. Toosi University of Technology. His research interests include computer networks, learning systems, soft computing, and information retrieval.



Mohammad Reza Meybodi received the B.S. and M.S. degrees in Economics from the Shahid Beheshti University in Iran, in 1973 and 1977, respectively. He also received the M.S. and Ph.D. degrees from the Oklahoma University, USA, in 1980 and 1983, respectively, in Computer Science.

Currently he is a Full Professor in Computer Engineering Department, Amirkabir University of Technology, Tehran, Iran. Prior to current position, he worked from 1983 to 1985 as an Assistant Professor at the Western Michigan University, and from 1985 to 1991 as an Associate Professor at the Ohio University, USA. His research interests include, channel management in cellular networks, learning systems, parallel algorithms, soft computing and software development.