IEIE Transactions on Smart Processing and Computing, vol. 9, no. 1, February 2020
https://doi.org/10.5573/IEIESPC.2020.9.1.058

DSLA: Defending against Selective Forwarding Attack in Wireless Sensor Networks using Learning Automaton

Mojtaba Jamshidi[1]*, Mehdi Esnaashari[2], Shahin Ghasemi[3]*, Nooruldeen Nasih Qader[4], and Mohammad Reza Meybodi[5]

[1] Department of Information Technology, University of Human Development, Sulaymaniyah, Kurdistan Region of Iraq
mojtaba.jamshidi@uhd.edu.iq
[2] Faculty of Computer Engineering, K. N. Toosi University of Technology, Tehran, Iran esnaashari@kntu.ac.ir
[3] Kermanshah University of Medical Sciences, Kermanshah, Iran shghasemi@kums.ac.ir
[4] Computer Science Department, University of Human Development, Sulaymaniyah, Kurdistan Region of Iraq
nooruldeen.qader@uhd.edu.iq
[5] Computer Engineering and Information Technology Department, Amirkabir University of Technology, Tehran, Iran
mmeybodi@aut.ac.ir

***** Corresponding Author: Mojtaba Jamshidi, Shahin Ghasemi

Received March 18, 2019; Revised November 28, 2019; Accepted December 17, 2019; Published February 28, 2020

** Regular Paper*

***Abstract*:** Selective forwarding attacks (SFAs) can harm the mission of critical applications such as military surveillance and forest fire monitoring. In these attacks, malicious nodes behave like normal nodes most of the time, but selectively drop sensitive packets, such as a packet that reports on the movement of opposing forces, and therefore, detection of this kind of attack is hard. In this paper, a fully distributed, dynamic, intelligent, lightweight algorithm based on learning automata is proposed in order to defend against the selective forwarding attack. In this algorithm, an overhearing mechanism, along with the learning automata model, is used to select secure routes for forwarding packets in a multi-hop routing algorithm. Each node is equipped with a learning automaton, which helps the node select the next hop for forwarding its data towards the base station. The proposed algorithm is simulated using J-SIM. Simulation results show the superiority of the proposed algorithm over existing algorithms, such as the single path forwarding algorithm, the multi-hop acknowledge–based algorithm, the multi-data–flow algorithm, the multi-path algorithm, and the neighbor watch system–based algorithm, in packet delivery rate, packet drop rate by malicious nodes, communications overhead, and energy consumption.

***Keywords*:** Sensor networks, Security, Selective forwarding attack, Learning automata

1. Introduction

The wireless sensor network (WSN) is a particular type of ad hoc network that consists of hundreds or thousands of small, inexpensive nodes that have the capability of sensing the surrounding environment with a certain goal (e.g. data processing, storage, exchanging data with other nodes) and that also have the ability to adapt to topology changes, etc. This problem becomes even more challenging considering the fact that, nowadays, sensor networks are widely used in military applications (such as border monitoring, and detecting the presence or movement of military vehicles or enemy forces). Due to the nature of wireless multi-hop transmission, sensor limitations (energy, storage, processing power, etc.), and the difficulty of monitoring and protecting every node against possible attacks, security is a very crucial challenge in WSNs [1, 2].

The selective forwarding attack is one of the network layer attacks that were discussed for the first time in [3]. In selective forwarding, a malicious node attempts to drop some of the received packets. If the malicious node drops all the received packets, it is easy to detect the attack, but if the malicious node begins selective forwarding (that is, dropping only a fraction of the received packets) detection will be harder and more challenging. This attack is especially effective if combined with Sybil and wormhole attacks, which can gather a lot of traffic through the malicious node [3]. These two attacks attract a lot of traffic, and if they launch an SFA at the same time, they can drop

more packets, and consequently, have more impact on the integrity of the network. Also, SFAs are typically most effective when the attacker is explicitly included in the path of a data flow. Note that it is conceivable for an adversary to overhear the flow passing through neighboring nodes in order to emulate SFAs by jamming or causing a collision with each forwarded packet of interest [4-6].

A possible approach to reducing the impact of SFAs is the use of a multi-hop acknowledgment–based scheme [7]. In this scheme, each intermediate node along the forwarding path is in charge of detecting malicious nodes. If an intermediate node detects misbehavior in its downstream or upstream nodes, it will generate an alarm packet and deliver it to the source node or the base station through multiple hops. The base station and the source node can then use a more complicated intrusion detection system (IDS) for further processing and reaction. The authors of this scheme assumed that routing and transport protocols, such as directed diffusion [9] and pump-slowly, fetch-quickly (PSFQ) [10], have attackers in the sensor nodes, and the scheme can function over these protocols. Generally, the defects in this scheme include the following. (1) The acknowledgement (ACK) messages and alert messages are forwarded along the original data routing path; if all the data are normal on the forwarding path, the running of the protocol is very smooth. However, as long as there is a malicious node on the data forwarding path, the malicious node can attack said routing path, which disables the protocol. (2) The number of ACK messages is too few to accurately locate malicious nodes. To save energy, previous research studies generated a minimal number of ACK messages, which are generated only by the checkpoint node. If the source node does not receive the expected ACK messages of the checkpoints, it cannot accurately locate malicious nodes. (3) There is a lack of scalability, and a slow reaction against the attack [11, 12].

In this paper, we propose a fully distributed, intelligent, lightweight, dynamic, and robust algorithm that can effectively bypass malicious nodes in SFAs. Also, the proposed algorithm is able to work even in the presence of mobile attackers or emulated SFAs, because it resists external selective forwarding or the same SFAs simulated through noise generation. The proposed algorithm, referred to as defending against selective forwarding attacks in wireless sensor networks using learning automaton (DSLA), uses the overhearing mechanism along with a learning agent to gradually learn secure routes for forwarding packets in a multi-hop routing protocol. DSLA has fast reactions, a high packet delivery rate to the base station, low communications overhead, low energy consumption, and easy adaptation to topology changes, as well as robustness against emulated SFAs (noise generation) and mobile attackers (nodes or malicious hardware that move in the network environment and simulate SFAs by producing noise).

The rest of this paper is organized as follows. The next section reviews the related work. Section 3 describes the learning automata. Section 4 presents a brief overview of the Localized Encryption and Authentication Protocol (LEAP), the key-establishment scheme, the network, and the attack model. Section 5 presents our algorithm, its phases, and some special cases, while Section 6 includes the performance evaluation and simulation results. Section 7 concludes the paper.

2. Related Work

Karlof and Wagner [3] were the first to discuss the SFA, and they also suggested multi-path routing to counter these types of attacks. In their method, the packets are routed through n completely separate paths from the source node towards the destination. Their method is resistant against the SFA until at most n nodes are not compromised. The drawbacks of their scheme are: (a) poor security resilience if there is at least one malicious node in each path, (b) not identifying malicious nodes, and (c) high energy consumption and communications overhead.

Xiao et al. [8] proposed a method for identifying suspicious nodes during SFAs. They actually improved their previous scheme [7] and named it the checkpoint-based multi-hop acknowledgment scheme (CHEMAS). In their scheme, a certain number of nodes are selected as checkpoint nodes in the forwarding path from the source node to the sink. As long as a checkpoint node receives a data packet, it will return an ACK to the upstream node. In addition, each node needs a one-way hash key chain for ensuring the authenticity of packets. Drawbacks of this scheme are (a) the need to have high amounts of storage space due to the use of one-way hash key chains for packet authentication; (b) more energy being consumed from sending ACKs and alert packets that include a one-way hash key; (c) no guarantee of reliable transmission of a packet if a packet is dropped; and (d) requiring nodes to be loosely time synchronized [12].

Kaplantzis et al. [13] proposed a centralized IDS based on support vector machines (SVMs) and used a sliding window mechanism to defend against black holes and SFAs. Anomaly detection is the base of their schema, and thus, anomaly detection observes user behavior. When a significant amount of unusual activity is observed, anomaly detection signals an intrusion. The drawbacks of this scheme are: (a) it only detects the execution of SFAs in the network, and it is unable to identify malicious nodes or discover alternate paths; and (b) the scheme is centralized, so it suffers from the single node–failure problem [11, 12].

Sun and Hsiao [14] proposed a multi-dataflow topology (MDT) method as a countermeasure to SFAs. This method was first presented in [15] to combat mobile jamming attacks. In the MDT scheme, the whole network is divided into different data topologies that make a sensor node belonging to one topology communicate and send information only through nodes under the same topology. This division can be done at different times. Generally, the division takes place on deployment. If not, then the nodes can randomly choose a topology after the deployment phase. In an MDT, if there is a malicious node in a data flow, the data can be rerouted to the sink successfully if there is a safely routed path in another data flow topology. However, the deficiencies in this type of scheme are (a) a

limited ability to resist attacks, and the scheme does not have the ability to identify compromised nodes; (if an attack occurs in all topologies, there will not be any that are defended against the attack); and (b) high energy consumption (due to multi-data transmission).

Watchdogs for SFAs and sinkhole attacks are the basis of the distributed IDS for sensor networks proposed by Ioannis and Dimitriou [16]. In creating the IDS, they employed the techniques of cooperative decision-making and specification-based rules. And for detecting SFAs in sensor networks, they used neighbor monitoring. The drawback of this scheme is that it is not efficient, because the final decision is taken after all the alerts are received from all the neighbors.

Lee and Cho [17] improved on an earlier multi-path routing method [3] for countering SFAs, and they proposed a fuzzy-based, reliable data-delivery algorithm. The improvement is that the number of transmission paths varies with the number of attackers. The number of paths for data delivery is determined by fuzzy logic with consideration for the energy level of the network and the number of malicious nodes. They assumed that the base station recorded (or can estimate) the number of malicious nodes, as well as the energy level of the network, in advance. In addition, all the nodes record their locations. The drawbacks of this scheme are (a) unreasonable assumptions; (b) the same limitations as the multi-path routing method; (c) high energy consumption due to redundant transmission of packets; and (d) it cannot identify compromised nodes.

Hai and Huh [18] proposed a centralized cluster-based detection algorithm to detect SFAs in WSNs. This scheme is based only on two-hop neighborhood information and an overhearing technique. The detection mechanism relies on the broadcast nature of sensor communications. Drawbacks with this scheme are (a) no proposed solution if the monitoring node or cluster head is compromised; (b) no countermeasures against SFAs, and (c) reliable data retransmission is not assured [11, 12]. Brown and Du [19] proposed a centralized cluster-based scheme for detecting an SFA in sensor networks. The scheme utilizes powerful high-end sensors, and is based on the sequential probability ratio test. The drawbacks of this scheme are (a) the single node–failure problem; and (b) no mechanism for reliable retransmission of dropped packets.

Lei et al. [20] proposed a polynomial-based mechanism against SFAs and a security scheme using redundant data to tolerate the loss of critical messages. The basic idea is to split the sensing data into parts, and to send these parts (instead of the original sensing data) to the sink. When the sink has received enough parts, it can parse the original sensing data, and if malicious nodes tamper with data, the sink can detect the tampered data. The drawbacks of this scheme are (a) the single node–failure problem; (b) dividing and processing the original data packet into small sizes leads to extra computational and storage overhead; and (c) high communications overhead (due to sending polynomial values to the sink) [11, 12].

Xin-sheng et al. [21] proposed a distributed scheme against the SFA, based on a hexagonal WSN mesh topology. This scheme utilizes neighbor nodes to monitor the transmission of an event packet and to detect SFAs by monitoring packet forwarding of two nodes in the transmission path, resending to the destination node those packets dropped by the attackers. The drawbacks of this scheme are (a) if there is any change in topology, it will affect the performance of the scheme; (b) no countermeasure is proposed in case the monitoring node is compromised; and (c) the scheme requires a global positioning system to acquire the locations of nodes, which makes the network costly [12].

Li et al. [22] proposed a sequential mesh test–based detection scheme for SFAs in WSNs. The nature of the scheme is centralized; it works for cluster-based sensor networks. The sensor node sends a dropped-packet report through another path to the cluster head if it does not observe the forwarding data message from the next-hop sensor node within a fixed interval. The cluster head runs the sequential mesh test–based detection scheme against a suspicious node after receiving the dropped-packet report. The drawbacks of this scheme are (a) detection of SFAs depends on the ratio of packets dropped by the malicious node to total packets; and (b) the single node–failure problem.

Park et al. [23] proposed an energy-efficient detection scheme for SFAs in WSNs. This scheme monitors the entire network based on the transmission time along the path transmitting each packet. It performs lazy detection for only the paths with the potential to have attack nodes. Hu et al. [24] proposed a security mechanism based on monitoring nodes, which are energy-heterogeneous nodes with the function of only monitoring (but not forwarding) in order to suppress malicious nodes' insider attacks.

Cui and Yang [25] focused on a node-reliability estimate to avoid selective forwarding. They take advantage of a modeling method by means of a sorting algorithm that can estimate the most suspicious nodes, and then avoids them in the routing, to analyze the forwarding behaviors of the nodes so that the selective forwarding can be deeply understood, and to finally establish a high-quality data-forwarding path. Liao and Ding [26] proposed a trust mechanism for identifying and removing malicious nodes that are established between the sending node and its one-hop neighbor nodes (a hybrid, continuous-strategy monitor–forward game), and it can effectively reduce the error rate in packet loss detection for unreliable radio channels.

Liu et al. [27] proposed a per-hop acknowledgment (PHACK)-based scheme for each packet transmission in order to detect SFAs. In this scheme, the sink and each node along the forwarding path generate an acknowledgment message for each received packet to confirm normal packet transmission. In this scheme, each ACK is returned to the source node along a different routing path. Zhou et al. [28] proposed a schema with three types of node: the cluster head (CH), the inspector node (IN), and member nodes (MNs). The IN monitors the CH's transmission in order to protect the cluster against SFAs; the CH forwards packets from MNs and other CHs, and randomly checks the IN to ascertain if it works properly. The MNs send the gathered data packets to the CH and evaluate the behaviors of the CH and IN based on their

own reputation mechanism.

Lee and Choi [29] proposed a resilient packet-forwarding scheme using a neighbor watch system (NWS) against maliciously packet-dropping nodes in sensor networks. Basically, this scheme employs single-path data forwarding, which consumes less power than multi-path schemes. A packet is forwarded along a single path towards the base station; however, the scheme uses multi-path data forwarding at the locations where the NWS detects the relay nodes' misbehavior. This scheme is based on the LEAP [30] protocol. The drawback of this scheme is high memory overhead (each node broadcasts its neighbor's table, and then stores the neighbor tables of its neighbors, and the watch nodes need to store packets around them for potential retransmission).

Mathur et al. [31] proposed two algorithms to defend against black hole and selective forwarding attacks for medical WSNs in the Internet of Things (IoT). The authors use cryptographic hashes to detect black hole attacks. Also, they use a neighbor watch system or NWS and threshold-based analysis to detect and correct selective forwarding attacks. Alajmi and Elleithy [32] proposed a three-layer scheme to defend against selective forwarding attacks. In the first layer, there is a pool of media access control (MAC) IDs that filter and match the traffic. In the second layer, a rule-based processing is run, which detects known attacks using rules. In the third layer, there is an anomaly detection system, which is the recognition of unknown attacks.

In this paper, we propose the new algorithm called DSLA to defend against SFAs in WSNs. DSLA bypasses malicious nodes in data paths by using an overhearing mechanism and a learning agent. Compared to the existing algorithms, DSLA has many advantages, such as (1) fast reaction when a packet drop occurs; (2) a high packet delivery rate; (3) it does not need extra packets (like ACKs) or sending data packets through multiple paths; (4) it is fully distributed and does not suffer from the single node–failure problem; (5) it offers adaptation to topology changes; and (6) it does not mark nodes as malicious, but just bypasses them in the data paths, so it has no false-detection rate.

3. Background

3.1 Learning automata

A learning automaton (LA) is a stochastic model operating in the framework of reinforcement learning. The automata approach to learning can be considered the determination of an optimal action from a set of actions. As shown in Fig. 1, an LA can be regarded as an abstract object that has a finite number of actions. It selects an action and applies that action to an environment. The environment evaluates the applied action and sends a reinforcement signal to the LA. The reinforcement signal provided by the environment is used to update the internal state of the LA. By continuing this process, the LA gradually learns the optimal actions to select, which leads to favorable responses from the environment [33-35].

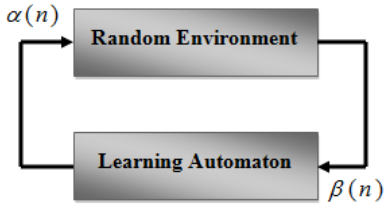


Fig. 1. Interaction of the LA and its environment [33].

An LA is a quintuple $<\alpha,\Phi,\beta,F,G>$ where $\alpha=(\alpha_1,\alpha_2,...,\alpha_r)$ is the set of actions that it must choose from, $\Phi=(\Phi_1,\Phi_2,...,\Phi_s)$ is the set of states, $\beta=(\beta_1,\beta_2,...,\beta_q)$ is the set of inputs, $G:\Phi\rightarrow\alpha$ is the output map, and determines the action taken by the LA if it is in the state Φ_j, and $F:\Phi\times\beta\rightarrow\Phi$ is the transition map that defines the transition of the state of the LA upon receiving input from the environment. The selected action at time instant *k*, denoted by $\alpha(n)$, serves as the input to the environment, which in turn emits a stochastic response, $\beta(n)$, at time instant *n*, which is considered the response of the environment to the LA. Based upon the nature of β, environments could be from one of three classes: P-, Q-, and S-models. The output of a P-model environment has two elements: success or failure. Usually, in P-model environments, a failure (or unfavorable response) is denoted by 1, whereas success (or a favorable response) is denoted by 0. In Q-model environments, β can take a finite number of values in the interval [0, 1] while in S-model environments, β lies in the interval [0, 1]. Based on response $\beta(n)$, the state of the LA, $\Phi(n)$, is updated and a new action is chosen at time instant (n+1).

An LA can be classified into two main families: the fixed-structure LA and the variable-structure LA. A variable-structure LA is defined by the quadruple $<\alpha,\beta,p,T>$ where α represents the action set of the LA, β represents the input set, $p=(p_1,p_2,...,p_r)$ represents the action probability set, and finally, $p(n+1)=T(\alpha(n)+\beta(n)+p(n))$ represents the learning algorithm. In time instant *n*, this LA operates as follows.

- Based on the action probability set, *up*(*n*), the LA randomly selects an action, $\alpha_i(n)$, and performs it in the environment.
- After receiving the environment's reinforcement signal, $\beta(n)$, the LA updates its action probability set based on the equations in (1) for favorable responses and the equations in (2) for unfavorable ones:

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

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

In the above equations, *a* and *b* are the reward and

penalty parameters, respectively. If $a = b$, the learning algorithm is called L_{R-P}; if $b << a$, it is called $L_{R\epsilon P}$; and if $b = 0$, it is called L_{R-I}.

LA with a Changing Number of Actions: In some applications, the number of possible or permissible choices, modeled by the actions of an LA, varies over time. For such applications, Thathachar and Bhaskar [36] proposed an LA with a changing number of actions in every time instant, n, but only a subset, V(n), of the actions of the LA is available for choice. The selection from action subset V(n) is made by an external system, which could determine permissible actions for a specific application. The LA chooses an action, $\alpha_i \in V(n)$, according to the probability distribution vector $\hat{p}(n)$ defined over V(n) by Eq. (3). The selected action is applied to a P-model environment and results in a binary response, $\beta(n)$. Then, the LA computes $\hat{p}(n+1)$ by using the equations in (4) (for favorable responses) or the equations in (5) (for unfavorable responses) [33-36].

$$p_i(n) = prob\left[\alpha(n) = \alpha_i \mid V(n) \text{ is set of active actions. } \alpha_i \in V(n)\right] = \frac{p_i(n)}{K(n)} \quad (3)$$

$$\begin{aligned} p_i(n+1) &= p_i(n) + a.(1 - p_i(n)) \quad \alpha(n) = \alpha_i \\ p_i(n+1) &= p_j(n) + a.p_i(n) \quad \alpha(n) = \alpha_i, \ \forall j \ \ j \neq i \end{aligned} \quad (4)$$

$$\begin{aligned} p_i(n+1) &= (1-b).p_i(n) \quad \alpha(n) = \alpha \\ p_i(n+1) &= \frac{b}{r-1} + (1-b)p_j(n) \quad \alpha(n) = \alpha_i, \ \forall j \ \ j \neq i \end{aligned} \quad (5)$$

Afterwards, the action probability vector of the LA is updated using the equations in (6):

$$\begin{aligned} p_j(n+1) &= p_j(n+1).K(n) \quad \text{for all j}, \alpha_j \in V(n) \\ p_j(n+1) &= p_j(n) \quad \text{for all j}, \alpha_j \notin V(n) \end{aligned} \quad (6)$$

The learning automata are considered model-free reinforcement learning methods where no model of the environment exists. In contrast, in model-based reinforcement learning methods, a model of the environment is assumed to be available. This model could be simplistic, noisy, incomplete, or even inconsistence (in some scenarios) with the real environment. However, it encompasses enough information to be substituted with the real environment, at least partially, in order to reduce the cost of real interactions (trial and error) of the agent with its real environment. It is clearly seen that if a suitable model is available, then model-based reinforcement learning methods perform better than model-free ones. However, in many real applications, such a model is not available. As an extreme point to this discussion, if a complete model of the environment is available, as assumed in dynamic programming solutions, the best action for the agent in any state can be computed, either computationally or by using iterative numerical methods.

3.2 Key-Establishment Scheme in LEAP

LEAP [30] supports the establishment of four types of key for each sensor node: an individual key shared with the base station, a pairwise key shared with its neighbor, a cluster key shared with its surrounding neighbors, and a group key shared by all the nodes in the network.

Individual Key: Every node has a unique key that it shares with the base station. This key is used for secure communication between the node and the base station. For example, a node can use its individual key to compute message authentication codes (MACs) for its sensed readings if the readings are to be verified by the base station.

Group Key: This is a globally shared key that is used by the base station for encrypting messages that are broadcast to the whole group. For example, the base station issues missions and send queries and interests.

Pairwise Shared Key: Every node shares a pairwise key with each of its immediate neighbors. Under LEAP, pairwise keys are used to encrypt communications that require privacy or source authentication.

Cluster Key: A cluster key is shared by a node and all its neighbors, and is mainly used for securing locally broadcast messages (for example, routing control information, or securing sensor messages), which can benefit from passive participation. Researchers have shown that in network processing techniques (including data aggregation), and passive participation is very important to reduce energy consumption in sensor networks. For instance, a node that overhears a neighboring sensor node transmitting the same reading as its own current reading can elect to not transmit its own reading. In responding to aggregation operations such as MAX, a node can also suppress its own reading if its reading is not larger than an overheard one. Clearly, for passive participation to be feasible, sensor nodes should be able to decrypt or verify some classes of messages (for example, sensor readings transmitted by their neighbors). This requires such messages to be encrypted or authenticated by a locally shared key. As such, LEAP provides each node with a unique cluster key shared with all its neighbors for securing messages. Its neighbors use the same key for decrypting or verifying messages.

4. System Assumptions and the Attack Model

In this section, we first present the network model and underlying assumptions for our scheme, and we then describe the attacker model. The main notations used in this paper are summarized in Table 1.

4.1 Network Model and Assumptions

We look at the WSN as a graph, $G = (V, E)$, where $V = \{v_1, v_2, ..., v_n\}$ is the set of nodes, and $E = \{e_{ij}\}$ is the set of communication links between nodes in the network, such that $|V| = \psi$ is the number of V's members, and $|E| = \xi$ is the number of E's members; .. if and only if $v_i \in V$ is adjacent to $v_j \in V$. The sensor nodes are divided

into two groups, source nodes (SNs) and forwarding nodes (FNs). It is assumed that all nodes are stationary, and each of them has a unique ID. We assume a two-dimensional area in which the SNs are manifestly distributed, and the FNs are randomly scattered. All of the nodes are homogeneous with transmission range R. It is assumed that the nodes communicate with one another via wireless radio channel, and they broadcast in omni-directional mode. When a node transmits a message, the message is received (i.e., heard) only by those nodes within the sender's communication range (designated hereafter as *neighboring nodes* or simply *neighbors*). The communication links are bi-directional, meaning that if node u can receive a message from node v, it can also send a message to node v. The SNs deliver their data to the base station throughout the FNs using a multi-hop route. It is assumed that the nodes are not tamper-resistant; that is, if they are compromised by an adversary, their confidential information will be revealed and the adversary is able to reprogram them and send them back as malicious nodes. It is also assumed that the LEAP scheme is used for setting up the keys among the nodes in the network.

4.2 Attack Model

The attack model used here is the one reported elsewhere [3, 7, 14]. When the sensor network is used in military applications, timely detection of events (such as armed forces movement) and the speedy forwarding of reports to the base station are important issues. But these processes can easily be corrupted by SFAs. In such attacks, malicious nodes may refuse to forward certain packets and simply drop them, ensuring they are not propagated any further.

As shown in Fig. 2, the adversary may attack in two ways: from inside the network via compromised nodes, or from outside the network by jamming the communication channels between uncompromised nodes. Fig. 2 also shows five basic ways in which malicious nodes can be distributed in a network for different tactical purposes. Fig. 2(a) shows a single malicious node located in the middle of a forwarding path. This node can selectively forward packets to the base station. Fig. 2(b) shows two or more malicious nodes chained along a forwarding path. This can make it difficult to detect dropped packets. Fig. 2(c) shows two or more malicious nodes that are not chained along a forwarding path. Fig. 2(d) shows a mobile/static, outside, malicious node that causes packets to be dropped by jamming. This can also make it difficult to detect dropped packets, especially when the outsider is mobile. Fig. 2(e) shows a number of compromised nodes surrounding a base station. This arrangement can be used to defeat the base station when the malicious nodes refuse to forward any packets at all.

Here, it is assumed that the adversary can compromise certain normal nodes in the network through physical capture or software bugs, reprogramming and spreading them into the network as malicious nodes; the adversary can also inject external malicious nodes into the network. We also assume that malicious nodes do not cooperate with one another. In the considered attack model, no

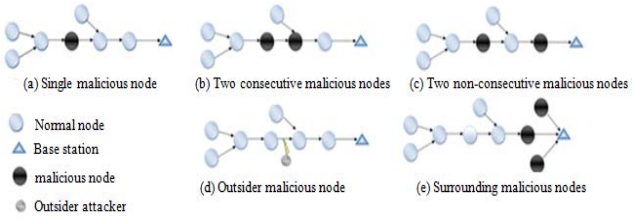


Fig. 2. Deployment of malicious nodes.

Table 1. Symbols.

Symbol	Description
u, v_i	Principals, such as communicating nodes.
$MinHopCount$	The minimum distance, in terms of hops, from a node to the base station.
$T_{OverHear}$	The time period in which a node should overhear packets forwarded by its next-hop node.
$SelectionSet$	Represents the action set of the LA.
P	Represents the action probability set of the LA.
T_{wait}	The time threshold below which a node listens to the channel to make sure that the next-hop node in the routing tree has sent its packets.
$Channel_{error}$	The error rate of the communication channel.
$N_{forward}$	The number of packets forwarded by each node to the base station.
M_{drop}	The minimum rate for dropped packets by each malicious node.
r	The number of the LA's actions.
R	The nodes' transmission range.
d	The average number of neighbors for a node.
T_C	The tolerable threshold for each node, meaning that if node u in its lifetime has penalized all of its next-hop neighbors (automata actions) more times than this threshold, node u will temporarily stop transmissions because malicious nodes have probably surrounded it.
LA_i	Represents node i's LA.
NH_i	Represents the next-hop node of node i towards the base station

constraint is imposed on the drop ratio of the packets.

The nature of wireless communications in WSNs causes the probability of the usual collisions and dropped packets to be high, especially for dense networks, as well as when the packet-sending frequency is high. This probability is even higher for the links close to the base station compared to the other links in the network. This is a major failure point for many of the existing algorithms, which causes some normal nodes to be mistakenly detected as malicious nodes.

In this paper, our goal is to design an algorithm capable of combating SFAs such that the following objectives will be fulfilled. First is to keep the communications, memory, and computational overhead, as well as energy consumption, as low as possible. Second is to have an algorithm that shows an instant reaction (unlike many existing algorithms [7, 8, 14]). Third is to keep the packet

delivery rate high and the dropped-packet rate low. Fourth, we need to have a fully distributed and dynamic algorithm that does not require any additional hardware, different capabilities for nodes, or special protocols in the transfer layer, such as PSFQ [10]. And the fifth objective is to have a very low probability of dropped packets as a result of malicious nodes.

4.3 Notations

We use the notations in Table 1 throughout the paper.

5. The Proposed Algorithm

In this section, we explain the proposed algorithm named DSLA. This approach can be applied to any type of multi-hop routing algorithm. Here, we use a multi-hop routing tree based on the shortest path, where the base station is considered the root of the tree. The proposed algorithm consists of three phases. The first phase is initialization, the second is configuration, and the third is data transmission. It is assumed that the first and second phases of the proposed algorithm are very short, and hence, the attackers do not have enough time to launch attacks during these two phases. We explain these three phases in following subsections, but before that, we introduce the packets used in DSLA.

5.1 Definitions of Packets

Three different packets are used in DSLA: reporting packets, distance vector (DV) packets, and alarm packets (Fig. 3).

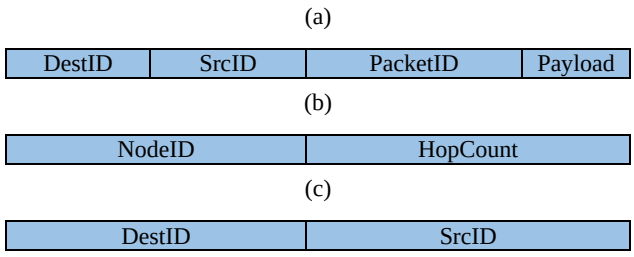


Fig. 3. The format of the packets: (a) the reporting packet, (b) the DV packet, (c) the alarm packet.

A reporting packet is generated at a source node when a special event (for example, tank movement) is detected, or in response to a query from a base station. After the reporting packet is generated, it is forwarded hop-by-hop from the source node towards the base station. This packet consists of the following fields: the identity of the destination node (*DstID*), the identity of the source node (*SrcID*), a unique identity for each packet (*Packet_ID*), and the intended data (*Payload*).

A DV packet is generated by each node during the first phase of DSLA to create the routing tree and determine the node's level in this tree. The suggested packet fields are shown in Fig. 3(b): the identity of the sender node (*NodeID*), and the distance from the node to the base station (*HopCount*).

An alarm packet is generated by a node that is surrounded by malicious nodes. The packet fields are *DstID* and *SrcID*, as seen in Fig. 3(c).

5.2 The First Phase: Initialization

In the proposed algorithm, each node v_i has a three-column routing table, which can be seen in Fig. 4: *NodeID* for neighboring node v_j, *NodeLevel* for node v_j in the routing tree, and *NumCrime*, which is described later in this section.

In addition, each node has a single field named *MinHopCount* that determines the minimum number of hops from that node to the base station; that is, the level of the node in the routing tree.

After deployment of the nodes in the environment, the base station as the root of the routing tree (the only node at level zero) generates a DV packet, encrypts it with its own K_{k0} clustering key, and then broadcasts it. The content of this packet is:

$$Base\ \ Station: \quad msg = \{NodeID = 0,\ \ HopCount = 0\}$$

Upon reception of a DV packet from a node v_j at a node v_i, node v_i performs the following.

1) Decrypts the received packet using K_{kv_j}

2) Adds a new row to its routing table with the following information:

$$\langle\ NodeID = v_j,\ \ NodeLevel = hopCount,\ \ NumCrime = 0\ \rangle$$

3) Sets level (v_i) to min rows in the routing table (*NodeLevel*) +1

4) Generates a new DV packet with the following information:

$$\langle\ NodeID = v_i,\ \ HopCount = level(v_i)\ \rangle$$

5) Encrypts the DV packet with K_{kv_i}

6) Broadcasts the packet in its vicinity.

NodeID	*NodeLevel*	*NumCrime*

Fig. 4. Routing table of nodes.

Here we specify the time required to perform this phase of the proposed algorithm. Assuming a network size of $X \times Y$, and the nodes' transmission range of R, we can calculate the network diameter per hop, L, using Eq. (7):

$$L = \left\lceil \frac{\sqrt{X^2 + Y^2}}{R} \right\rceil \tag{7}$$

So, the DV packet of the base station should travel d_h hops. The required time in each hop, T_{hop}, can be calculated with Eq. (8):

$$T_{hop} = Q_{delay} + T_{delay} + P_{delay} \tag{8}$$

where Q_{delay} is the time required to queue and process

(decrypt, encrypt, etc.) the DV packet; T_{delay} is the transmission delay, and P_{delay} is the propagation delay, which are calculated from Eqs. (9) and (10), respectively:

$$T_d = \frac{packet\ \ size\ \ (bit)}{bandwidth} \tag{9}$$

$$P_d = \frac{distance}{speed\ \ in\ \ medium} \tag{10}$$

Hence, the total time required to execute the first phase of the proposed algorithm can be estimated by $L \times T_{hop}$.

5.3 The Second Phase: Configuration

Under DSLA, each sensor node v_i is equipped with an LA that includes a number of actions: LA_i. At the beginning, each node v_i examines its routing table and finds those rows where *NodeLevel* is less than its own level (v_i) and adds the rows to a set called *SelectionSet.* This set, in fact, is the action set of the LA. That is, each action of LA_i corresponds to the selection of one of the neighbors of v_i. In the beginning, all of the actions of the LA are active, i.e. $V(n) = SelectionSet$. and have equal probabilities calculated from the Eq. (11):

$$P_{i.k} = \frac{1}{r}.$$
$$\text{where } r = \left|SelectionSet(v_i)\right|; \forall \alpha_k \in SelectionSet(v_i) \tag{11}$$

Then, from the LA (that is, LA_i) one action (α_k) is selected at random. This action indicates one of the neighbors of v_i as the next hop from v_i in the path towards the base station. We refer to the selected neighbor as NH_i hereafter.

In addition, v_i sets *PktCount* (NH_i) to zero. This variable is used later as a reinforcement signal for LA_i.

This phase of the proposed algorithm does not need any communication; it just needs some in-memory operations on the order of $O(d)$ and so, takes place very quickly.

5.4 The Third Phase: Data Transmission

When node v_i (as a source node) has data to send towards the base station, it generates a report packet with the following information:

$$< NextHopID = NH_i.SrcID = v_i.PacketID$$
$$= a\ new\ unique\ identity.\ Payload$$
$$= Data >$$

This report packet is then encrypted with cluster key K_{kv_i} and sent to node NH_i; v_i then increments the value of *PktCount* (NH_i) by 1. Each node v_j upon reception of a report packet from node v_i performs the following steps.

Algorithm 1. Updates the action probability vector of LA_i

For $i = 1$ *to* $PktCount(NH_i) - 1$ *do*
begin
gift reward to action α_k *corresponding to* NH_i *according to equation*(4)
end
penalizes action α_k *corresponding to* NH_i *according to equation*(5)

1) Decrypts the received packet using K_{kv_i}

2) Replaces the *NextHopID* field in the packet with its own next hop: NH_j

3) Encrypts the modified packet with K_{kv_j}

4) Sends out the encrypted packet to NH_j

5) Sets *PktCount* (NH_j) = *PktCount* (NH_j)+1

It should be noted that if the attackers compromise a node in order to launch an SFA, then the node might drop one or some of the received packets instead of forwarding them towards the base station. Under DSLA, we make use of the overhearing mechanism to detect such malicious packet drops. To this end, each sensor node v_i after sending out a report packet to its next hop, NH_i, monitors the wireless channel for a set duration, T_{wait}, to make sure the packet will be forwarded by NH_i.

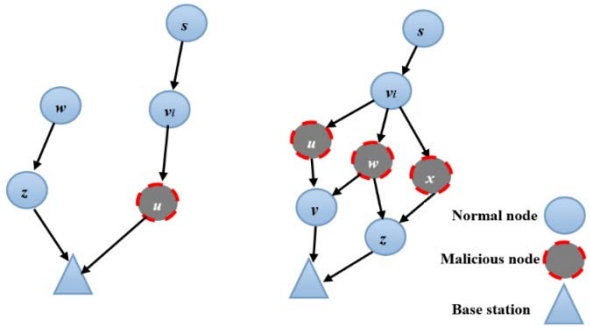
If v_i detects that NH_i does not forward the packet, then it records the malicious behavior by NH_i. In this case, v_i performs the following steps, referred to as the malicious-behavior procedure.

- Update the action probability vector of LA_i using the procedure given in Algorithm 1.
- Delete the action corresponding to NH_i from the active set of LA_i
- Set *Old-*NH_i = NH_i
- LA_i selects a new action, α_k, randomly according to its action probability vector. This new action becomes the new next hop for v_i; that is $NH_i = \alpha_k$
- Set the *PktCount* (NH_i) to zero
- Insert the action corresponding to *Old-*NH_i into active action set LA_i

Since overhearing can consume a lot of energy, this mechanism is only performed every $T_{OverHear}$ packets.

The important issue in the proposed algorithm is that no node will be marked as a malicious node, and no messages regarding the detection of malicious nodes will be broadcast to the network. Instead, the LA will decrease the probability of selecting action with those malicious nodes so such nodes are very rarely selected as the next hop in the routing tree. This characteristic, especially when the adversary simulates an SFA using an external mobile node (and the generation of jamming) is very suitable. For example, consider a scenario in which an external mobile adversary is adjacent to normal node v and interferes with the packets being forwarded by jamming them. As a result, the neighbor nodes that selected node v as the next hop will consider node v malicious, and will not send packets to it. Now, if the external mobile adversary moves to another location in the network, taking into account the nature of the next-hop selection from the LA, node v will have a chance to once again be selected by its neighbors as a reliable next hop.

Therefore, as time passes, the LA learns the suitable



(a) (b)

Fig. 5. Two possible topologies of the sensor network.

next hops for sending packets to the base station. This process takes place distributively and dynamically by all LAs, and prevents malicious nodes from being along the data paths.

5.5 Special Cases

The first scenario: If the network is very sparse, like the one given in Fig. 5(a), there is a high probability that a node such as v_i at the lth level of routing has only one neighbor node, like u, at the (l-1)th level. This means that LA_i will have only one action for selection. Under such situations, if node u is compromised by an adversary and tries to maliciously drop packets, although v_i can recognize node u as a malicious node, it cannot select another node for the next hop. To overcome such a problem, node v_i can forward its packets through neighbor nodes located on the lth level of the routing tree (for example, in Fig. 5(a), through node w towards the base station).

To this end, once a node in the lth level of the routing tree finds no neighbors on the (l-1)th level for forwarding packets towards the base station, it inserts the neighbors from the lth level of the routing tree into its *SelectionSet* and reconfigures its LA accordingly. If no neighbor exists at the lth level of the routing tree, then v_i generates an alarm packet and sends it to the neighbors from which it receives report packets. Thus, neighbors stop sending their packets to v_i, and choose a different path towards the base station. When a node receives an alarm packet from its next-selected hop, it assumes the next hop is a malicious node and executes the procedure used when malicious behavior is detected (the malicious-behavior procedure explained in Section 5.4).

The second scenario: Consider the situation shown in Fig. 5(b), where node v_i (located on the lth level of the routing tree) has more than one neighboring node at the (l-1)th level, but all of them are labeled malicious. In other words, multiple malicious nodes surround one node. Under such circumstances, v_i continuously penalizes its next hop, and changes from one malicious node to another as the next hop. To overcome such a problem, v_i records the number of times that it penalizes each of its next-hop neighbors in the *NumCrime* field of the routing table.

Whenever node v_i (located on the lth level) determines that the value of this field is greater than a preset threshold, T_c, for all of its neighbors located at the (l-1)th level, it then has no route towards the base station on the (l-1)th level of the routing tree. This is identical to the first scenario. Therefore, v_i can perform the steps described there to overcome the situation.

Lemma 1: Let v_i be a node at the lth level of the routing tree; let r be the number of neighbors of v_i at the (l-1)th level of the routing tree, and let N_f be the number of packets forwarded by v_i towards the base station through these neighbors. If $Channel_{error}$ is the error rate of the wireless channel, and if M_{drop} is the minimum drop probability of the packets by malicious nodes, then T_c, the minimum value for the *NumCrime* field above, with which v_i can be sure all its neighbors in the (l-1)th level of the routing tree are malicious, is:

$$T_c = ((1 - channel_{error}) \times \frac{N_{forward}}{r} \times M_{drop}) + (\frac{N_{forward}}{r} \times channel_{error}) \quad (8)$$

The proof of this *Lemma* is in Appendix A.

6. Performance Evaluation and Simulation Results

In this section, we first evaluate the communications and memory overheads of the proposed algorithm. Then, we evaluate the performance of the algorithm through simulations, comparing it with other existing algorithms such as the single-path, the multi-path–based [3], ACK-based [7], MDT [15], and NWS [29] algorithms.

6.1 Communications Overhead

Algorithms given in [3, 7, 15] use plenty of ACK packets for assuring the successful delivery of a single packet to the base station in the presence of SFAs. The algorithm given in [29] does not use ACK packets; instead, if it observes malicious behavior, it uses multi-path transmission, which causes an increase in communications overhead. On the other hand, in the proposed algorithm each node monitors its next hop for delivering the packets by overhearing the channel to make sure the next hop sends its packets. This algorithm does not need any multi-path transmission or ACK packets. Therefore, the mean communications overhead of the proposed algorithm for delivering packets is less than the algorithms in [3, 7, 15, and 29].

6.2 Memory Overhead

In a network where each node has d neighbor nodes, on average, the memory overhead of the algorithm [29] is equal to $O(d^2)$. However, with the proposed algorithm, each node should keep the routing table in a $3d$ memory space, as well as the *SelectionSet* and the action probability vector in d memory spaces. Therefore, the total memory overhead of the proposed algorithm is $5d$, and is on the order of $O(d)$.

6.3 Simulation Results

In order to evaluate the efficiency of the proposed algorithm and compare it with other algorithms, we used the J-SIM simulator [37]. J-SIM is an open source and component-based network simulator developed entirely in Java. It provides a real-time process-based simulation. The main benefit of J-SIM is its considerable list of supported protocols, including a WSN simulation framework with a very detailed model of WSNs, and an implementation of localization, routing, and data diffusion WSN algorithms. J-SIM models are easily reusable and are interchangeable, offering maximum flexibility [37-39]. Already, many works [1, 40-43] exist in various WSN fields that are simulated using J-SIM.

In our simulations, we used a sensor field of 100 m^2 in which 300 nodes were randomly distributed. We positioned a stationary base station at location (50, 30), with 20 stationary source nodes at coordinates of (x, 100), where x changes from 0 to 100 at intervals of five meters.

Each source node generates one report packet every five seconds. Each sensor node has a constant transmission range of R = 10 m. All malicious nodes were randomly selected by other nodes (i.e., excluding the base station and the source nodes). In order to avoid detection, malicious nodes drop only part of the packets passed by them. Therefore, in all the experiments, except for experiment No. 5, we assumed that the probability of a packet drop by a malicious node is 50%. The channel error rate was set to its default value in J-SIM. For all algorithms, the carrier-sense multiple access (CSMA) protocol in the MAC layer was used. We assumed the initial energy for each node to be five joules. The default values of parameters for the reward (a), the penalty (b), and $T_{OverHear}$ in the proposed algorithm are 0.001, 0.00001, and 1, respectively. In the proposed algorithm we employed the $L_{R\epsilon P}$ model of learning automata in which $b<<a$ (0.00001<<0.001). Each simulation ran for 10,000 seconds. Also, each simulation experiment was conducted using 50 different network topologies, and each result was averaged over 10 runs from any of the 50 topologies. The following metrics were used in the simulations.

- **Packet delivery rate:** The percentage of packets generated by the source nodes and that reached the base station.
- **Packet drop rate:** The percentage of packets dropped by malicious nodes and that never reached the base station.
- **Relative communications overhead:** Measures the ratio of the total communications overhead in a system that incorporates our (or each) detection scheme against a system that suffers no SFAs (which we call the base system) [8].
- **Average remaining energy in nodes:** This metric measures the average remaining energy in all of the nodes during the network lifetime.

To show the efficiency of the proposed DSLA, we compared the simulation results from the algorithm with five other algorithms, which are shown in Table 2. Also, in our simulations, we adjusted the security parameters of the other algorithms, as shown in Table 2.

Table 2. List of other algorithms compared to the proposed algorithm.

Algorithm	Security parameters
Single Path Forwarding (base system)	-
Multi-Path–based [3]	Uses at most 5 distinct paths for leading each packet toward the base station
MDT [15]	Uses two distinct data topologies
ACK-based [7]	$ACK_{span} = 2$, $ACK_{TTL} = 4$, $t = 1$
NWS [29]	-

Experiment 1: This experiment was conducted to evaluate the performance of the proposed algorithm in comparison to four other algorithms in terms of packet delivery rate, packet drop rate, and relative communications overhead. For this experiment, we set $T_{OverHear} = 1$, $a = b = 0.0001$. Fig. 6 presents the results of this experiment. It is clear from the results in Fig. 6(a) that the packet delivery rate under the DSLA algorithm is higher than the other algorithms. The main reason is that DSLA is fully distributed and behaves intelligently such that, when any misbehaving node is observed, it instantly eliminates that node from data paths. For example, with the existence of 80 malicious nodes in the network, the packet delivery rate under DSLA was about 90%, whereas for the other algorithms, it was lower than 70%. Another example is when the number of malicious nodes increases to one-third of the total number of nodes. The packet delivery rate with DSLA was more than 70%, but for the other algorithms, it was less than 40%.

Furthermore, the results in Fig. 6(b) indicate that when the number of malicious nodes in the network is less than 25% of the total number of nodes, the packet drop rate was less than 3% (here, 75 nodes). When the number of malicious nodes reached one-third of the total number of nodes (here, 100 nodes), this metric was less than 20%. However, the packet drop rate for the other algorithms was more than 35% in the first case, and was more than 50% in the second case. But in the multi-path and MDT algorithms, misbehaving nodes cannot be detected, and therefore, the rate of packet drops due to malicious nodes was high, especially when the number of malicious nodes was high. Also, in the ACK-based algorithm, the source nodes and the base station carry out detection of the misbehaving nodes, selecting alternate paths. Therefore, the reaction is too slow, which results in many dropped packets before selecting an alternate, secure path.

By definition, the relative communications overhead of single path forwarding (base system) is equal to 1. The communications overhead under multi-path forwarding is n times as much as the base system, where n is the number of paths in multi-path forwarding. Also, the relative communications overhead of the MDT algorithm is at least m times as much as the base system, where m is the number of distinct data flow topologies. This is because, in this algorithm, “every data flow covers the whole desired field,” and source nodes dispatch the packets to all their neighbors. Therefore, each packet flows towards the base

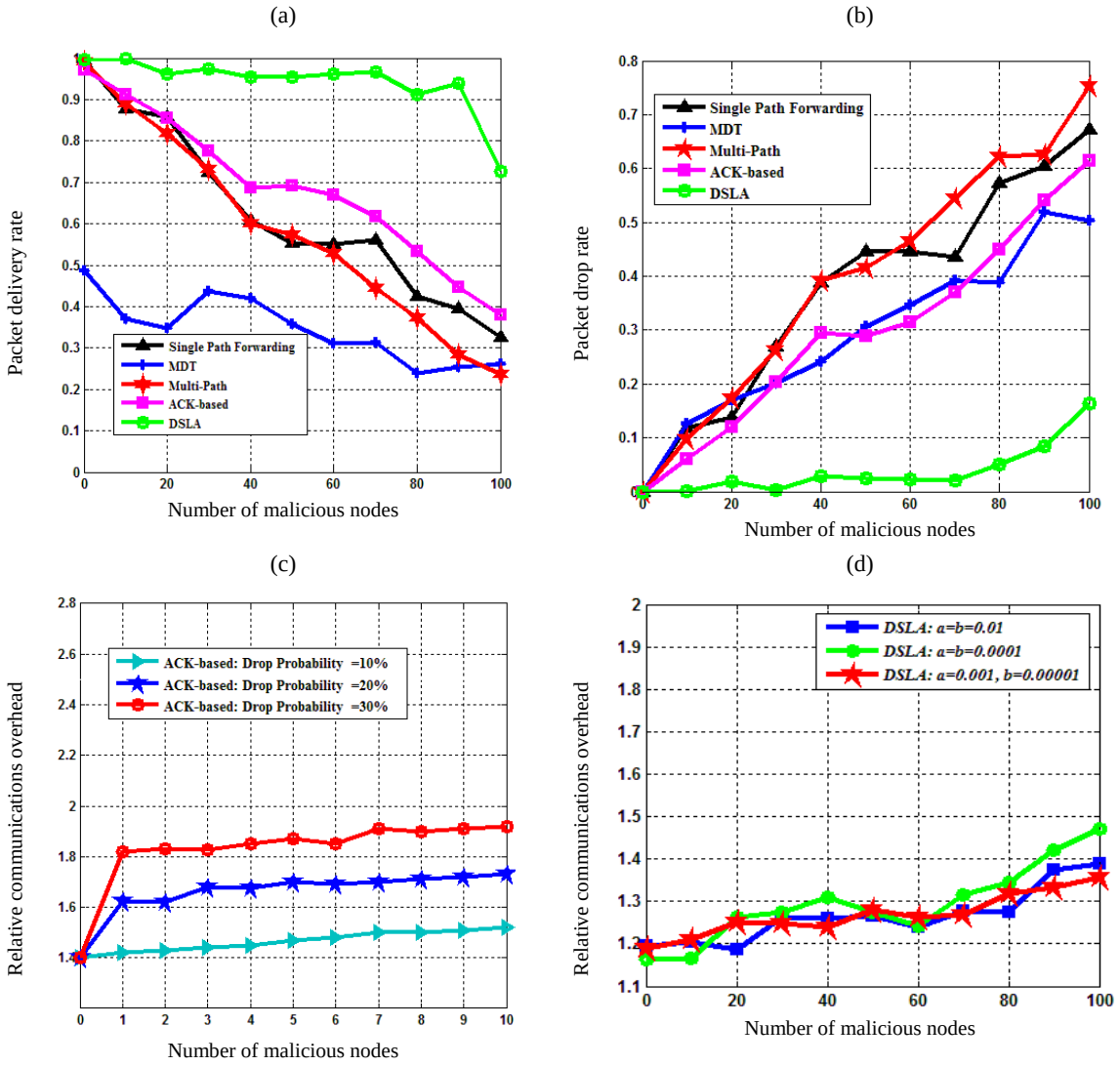


Fig. 6. Comparison of the proposed algorithm and the other algorithms in term of (a) the packet delivery rate, (b) the packet drop rate, (c) the relative communications overhead of the ACK-based algorithm [7], (d) the proposed algorithm for $T_{OverHear}=1$ and different Las.

station through not only *m* distinct data flows but also possibly multiple separate paths within a data flow. This metric for the ACK-based algorithm [7] was evaluated by simulation. In the conducted simulation, there was only one source node, and the number of malicious nodes changed within the range [0, 10]. The results of this experiment, as shown in Fig. 6(c), indicate that the relative communications overhead of the ACK-based algorithm when the drop probability was 0.1 was more than 1.4, and when the drop probability was 0.3, it reached 1.8. The results also show that the relative communications overhead of the ACK-based algorithm increased considerably when the number of malicious nodes and/or the drop probability increased.

The relative communications overhead of the proposed algorithm under more severe conditions, where the number of malicious nodes was 0, 10, 20, and 100, and the drop probability was 50%, can be seen in Fig. 6(d). It is obvious from Fig. 6(d) that the relative communications overhead of the proposed algorithm was less than the ACK-based algorithm. And the reason is that each node in the ACK-based algorithm must transmit a lot of ACK packets in addition to reporting packets.

Experiment 2: In this experiment, we examined the effect of the reward (*a*) and the penalty (*b*) parameters of the LA on the proposed algorithm in terms of packet delivery rate and packet drop rate. For this experiment, we set $T_{OverHear}=1$. The results of this experiment are shown in Fig. 7.

The results show that if the L_{RP} LA is used, the values for the *a* and *b* parameters do not affect the performance of DSLA significantly. However, if L_{ReP} LA is used, better results (compared to using L_{RP} LA) will be achieved. When the value of the reward is greater than the value of the penalty, the action probability values corresponding to normal next-hop nodes increase faster than the decreasing rate of the action probability values corresponding to malicious nodes. Therefore, the possibility for selection of normal nodes as the next hop increased, and consequently, the packet delivery rate increased, as seen in Fig. 7(a), and

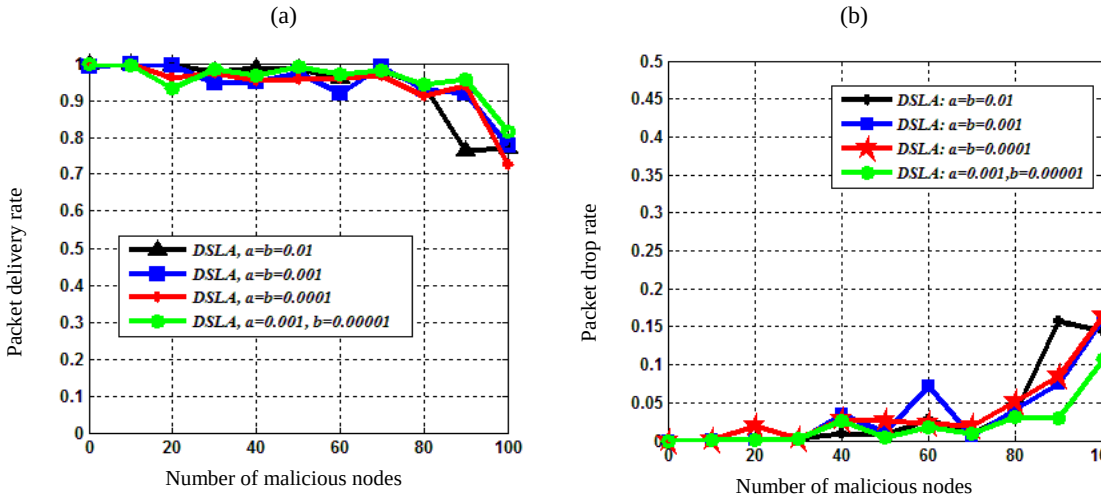


Fig. 7. The effect of the reward and penalty parameters on (a) the packet delivery rate, (b) the packet drop rate of the proposed algorithm.

following it, the packet drop rate decreased, as seen in Fig. 7(b). Also, the relative communications overhead increased, as illustrated in Fig. 6(d).

The effect of the reward (a) and the penalty (b) parameters of the LA on the proposed algorithm in terms of relative communications overhead are shown in Fig. 6(d).

Experiment 3: The objective of this experiment was to evaluate the effect of $T_{OverHear}$ on the performance of the proposed algorithm in terms of packet delivery rate, packet drop rate, and relative communications overhead. $T_{OverHear}$ is a security parameter, and the experiment results in Figs. 8(a) and (b) show that the smaller $T_{OverHear}$ is, the faster the discovery of malicious modes. Therefore, fewer packets are dropped by such nodes. Accordingly, the rate of delivering packets to the base station increased. Thus, the relative communications overhead under the proposed algorithm decreased, as seen in Fig. 8(c). It is possible to take into consideration a fixed value for $T_{OverHear}$ in all of the nodes, or each node can pick a random value between *minThreshold* and *maxThreshold* for it. Fixing the value of $T_{OverHear}$ for all the nodes decreases security because malicious nodes might guess $T_{OverHear}$ and will not drop packets during the overhearing periods. Therefore, for the proposed algorithm, we recommend randomly changing the value of $T_{OverHear}$ from time to time.

Experiment 4: In this experiment, we evaluated the packet delivery rate of DSLA, NWS, and the base algorithm, where there is only one source node in the network and the malicious nodes are strategically scattered in the desired field. In this experiment, the number of malicious nodes was varied from 0 to 50. Malicious nodes were located in a 50 m × 50 m area. Thus, malicious nodes were strategically placed between the base station and the source node. In the experiment, malicious nodes dropped all the relayed packets; that is, the packet drop probability was 1. Fig. 8(d) shows the result of this experiment.

As is obvious from the results, the packet delivery rate with the proposed algorithm was more than 99%, whereas this metric for NWS and the base algorithm was 60% and 0%, respectively, when the number of malicious nodes was 50. The results indicate that the proposed algorithm is efficient, even under such severe situations.

Experiment 5: This experiment was conducted to evaluate the effect of the packet drop probability on the performance of the proposed algorithm. Considering the nature of wireless transmissions in sensor networks, even without the existence of malicious nodes, some of the packets are dropped because of channel errors. Therefore, in most of the existing algorithms, in order to distinguish between packets dropped maliciously and those dropped because of channel errors, the rate of packets being dropped by malicious nodes (the drop probability) is assumed to be much higher than the rate of channel errors. For example, in NWS, the simulation results are presented under the assumption that drop probability is equal to 1.

In this experiment, the number of malicious nodes was 50. We changed the drop probability from 0.1 to 1, and the results of this experiment (in terms of packet drop rate and relative communications overhead) are shown in Figs. 9 and 10, respectively. As expected, when drop probability is equal to 0.1, the packet drop rate reached 1%, but as the drop probability increased, the rate of this metric decreased (to below 0.5%). The results of this experiment show that even if malicious nodes have a low probability of dropping packets, the proposed algorithm performs well, and it detects the misbehaving nodes and eliminates them from data paths. From Fig. 9, we conclude that changes in the drop probability parameter cannot have much effect on the relative communications overhead of the proposed algorithm. Also, we can conclude that relative communications overhead under the proposed algorithm is less, compared to the NWS algorithm. This is because under NWS, when a packet is dropped, it is possible that the packet is sent again through multiple distinct paths towards the base station, which increases the relative communications overhead of this algorithm. If we have the topology in Fig. 11, when node u sends a packet to its next hop (that is, node m) the packet will be overheard by the common neighbors of nodes u and m (that is, watch nodes

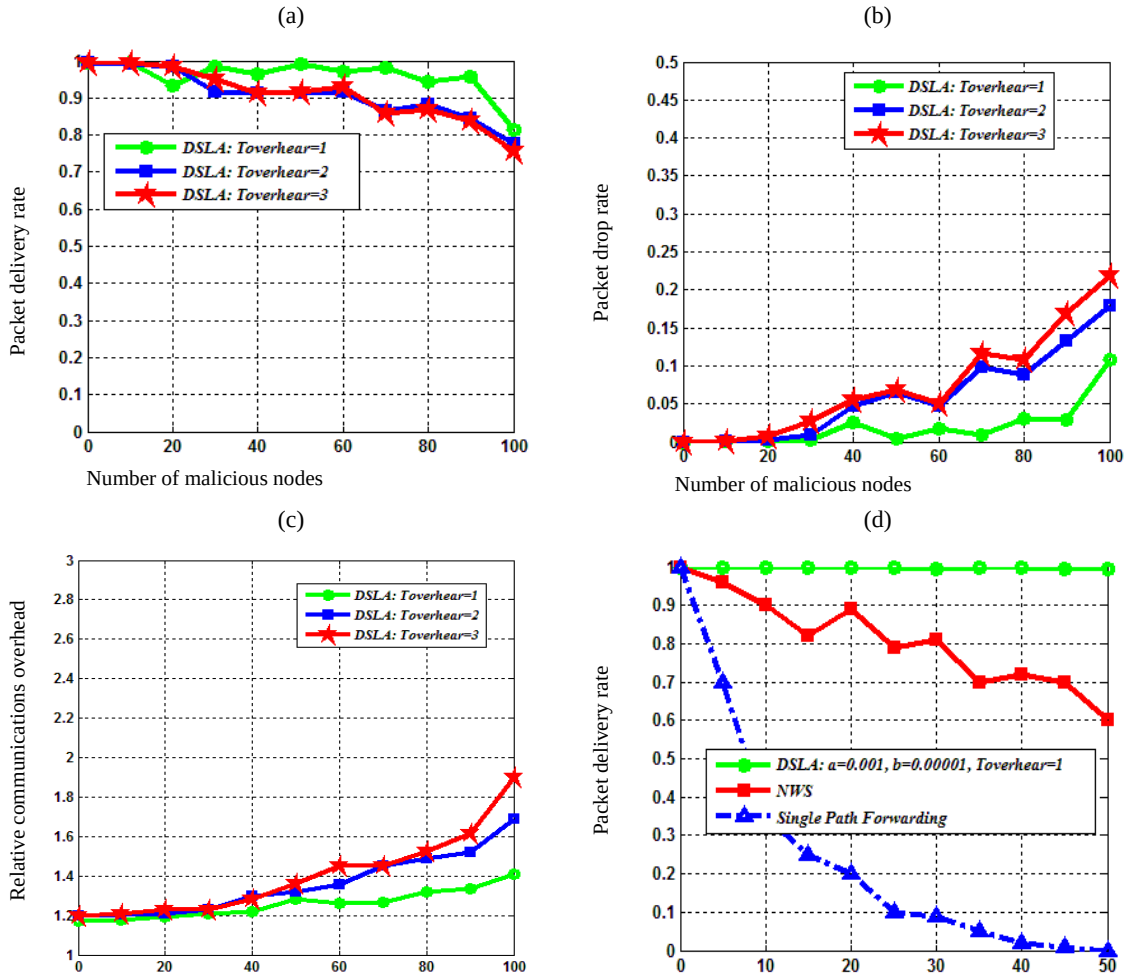


Fig. 8. The effect of the $T_{OverHear}$ parameter on (a) the packet delivery rate, (b) the packet drop rate, (c) the relative communications overhead of the proposed algorithm for a = 0.001 and b = 0.00001, (d) a comparison of the proposed algorithm with NWS and the base algorithm in terms of packet delivery rate.

z, *v*, and *w*). Now, if node *m* drops the packet, the watch nodes, and node *u* itself, attempt to forward it again. In the NWS algorithm, in fact, several nodes will escort a single packet towards the base station, and this will increase computation and memory overhead in addition to communications overhead. On the other hand, with DSLA, only node *u* monitors the activities of its next hop (node *m*), and if it finds malicious behavior, *u* will change the next hop to any of the watch nodes (*z*, *w* or *v*) and will resend the packet; that is, only one copy of the packet will be resent. Therefore, it is obvious that the relative communications overhead under DSLA is less than the NWS algorithm.

Experiment 6: In this experiment, we compared DSLA with the other algorithms in terms of energy consumption by the nodes. In this experiment, we had 50 malicious nodes, $T_{OverHear} = 1$, $a = 0.001$, and $b = 0.00001$. Fig. 12 shows the result of this experiment for the time period between 8800 and 10,000 seconds. As is obvious from Fig. 12, the average remaining energy in the nodes under the proposed algorithm is less than the base algorithm and more than all other algorithms. These results are very clear, because the MDT and multi-path algorithms forward a single packet through multiple paths towards the base station, and so, the energy consumption to deliver a packet from the source node to the base station is very high. Also, the ACK-based algorithm consumes a lot of energy by sending ACK packets, whereas DSLA does not use multi-path forwarding or ACK packets, and only uses the overhearing mechanism. Thus, overhearing packets (in comparison to sending packets) consumes far less energy. Since the base algorithm does not resend the dropped packets, the number of its transmissions is extremely low. Therefore, under the base algorithm, nodes consume a significantly lower amount of energy, in comparison to the proposed algorithm.

7. Conclusion

In this paper, we proposed DSLA, a fully distributed, dynamic, intelligent, and lightweight algorithm based on LA for protecting sensor network traffic against SFAs. Under DSLA, the overhearing mechanism, along with the LA model, is used to select secure routes for forwarding packets in a multi-hop routing protocol. The proposed

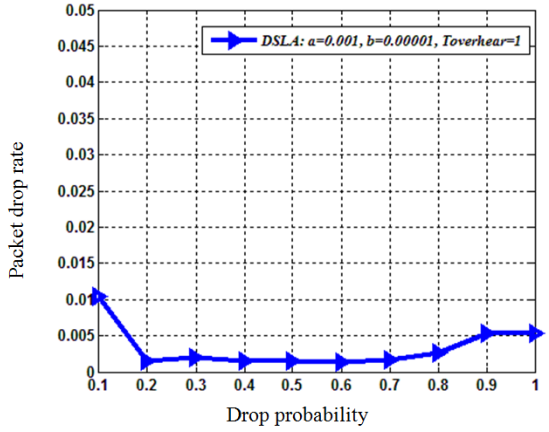


Fig. 9. The effect of the drop probability for malicious nodes on the packet drop rate under the proposed algorithm.

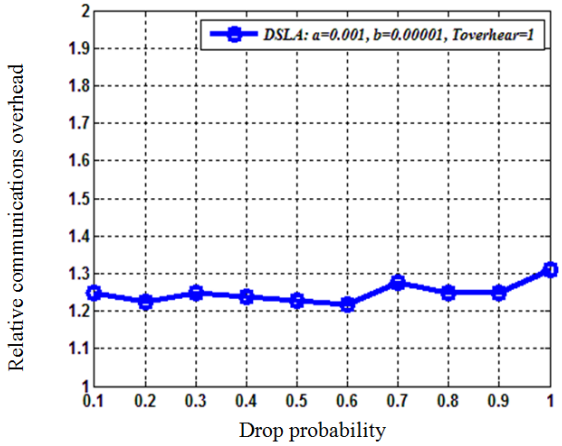


Fig. 10. The effect of the drop probability for malicious nodes on the relative communications overhead of the proposed algorithm.

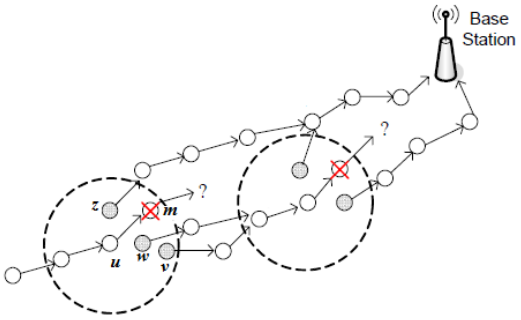


Fig. 11. An example of packet forwarding under the NWS algorithm [29].

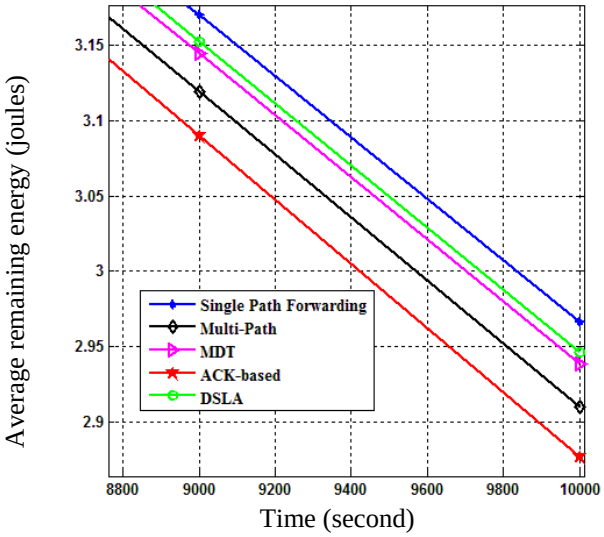


Fig. 12. Comparing the proposed algorithm and the other algorithms in terms of average remaining energy.

algorithm was simulated in different scenarios using J-SIM. Simulation results show the superiority of the proposed algorithm over existing algorithms, such as the single path forwarding algorithm, the multi-hop ACK-based algorithm, the multi-dataflow algorithm, and the multi-path algorithm, in terms of packet delivery rate, packet drop rate by malicious nodes, communications overhead, and energy consumption.

Appendix A: Proof of Lemma 1 (calculating the T_C threshold):

To calculate T_C, assume node u has sent $N_{forward}$ reporting packets to r existing neighbors (*SelectionSet*). If all the nodes in *SelectionSet* are malicious (the theory of the problem), in this case, the LA will select any of these malicious neighbors, alternately and almost equally, as the next hop from node u. Therefore, node u will send N_s packets to any of these neighbors:

$$N_S = \frac{N_{forward}}{r} \tag{A.1}$$

Out of this number of packets (N_s), T_{C1} packets will be dropped because of channel errors, and T_{C2} packets will be dropped by malicious nodes:

$$T_{c1} = N_S \times Channel_{error} \tag{A.2}$$

$$T_{c2} = (1 - channel_{error}) \times N_s \times M_d \tag{A.3}$$

where $channel_{error}$ is the rate of channel errors, and M_{drop} is the minimum rate of dropped packets by malicious nodes. Consequently, the tolerable threshold for dropped packets by each node, T_C, equals

$$T_C = T_{c1} + T_{c2} \tag{A.3}$$

$$T_C = ((1 - channel_{error}) \times \frac{N_{forward}}{r} \times Min_{drop\_rate}) + (\frac{N_{forward}}{r} \times channel_{error})$$

References

[1] Jamshidi, M., Zangeneh, E., Esnaashari, M., Meybodi, M. R. (2017). A lightweight algorithm for detecting mobile Sybil nodes in mobile wireless sensor networks. Computers & Electrical Engineering, 64, 220-232. Article (CrossRef Link)

[2] Jamshidi, M., Darwesh, A. M., Lorenc, A., Ranjbari, M., Meybodi, M. R. (2018). A Precise Algorithm for Detecting Malicious Sybil Nodes in Mobile Wireless Sensor Networks. IEIE Transactions on Smart Processing & Computing, 7(6), 457-466. Article (CrossRef Link)

[3] Karlof, C., Wagner, D. (2003). Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures. Ad Hoc Networks, 1(2), 299-302. Article (CrossRef Link)

[4] Diaz A., Sanchez, P. (2016). Simulation of attacks for security in wireless sensor network. Sensors, 16(11), 1-27. Article (CrossRef Link)

[5] Bhushan, B., Sahoo, G. (2018). Recent Advances in Attacks, Technical Challenges, Vulnerabilities and Their Countermeasures in Wireless Sensor Networks. Wireless Personal Communications, 98(2), 2037-2077. Article (CrossRef Link)

[6] Jamshidi, M., Zangeneh, E., Esnaashari, M., Darwesh, A. M., Meybodi, M. R. (2019). A Novel Model of Sybil Attack in Cluster-Based Wireless Sensor Networks and Propose a Distributed Algorithm to Defend It. Wireless Personal Communications, 105(1), 145-173. Article (CrossRef Link)

[7] Yu, B., Xiao, B. (2006). Detecting selective forwarding attacks in wireless sensor networks. Int'l Workshop on Security in Systems and Networks, 1-8.

[8] Xiao, B. Yu, B., Gao, C. (2007). CHEMAS: identify suspect nodes in selective forwarding attacks. Journal of Parallel and Distributed Computing. 67(11), 1218-1230. Article (CrossRef Link)

[9] Intanagonwiwat, C., Govindan, R., Estrin, D. (2000). Directed diffusion: A scalable and robust communication paradigm for sensor networks. ACM MobiCom. 56-67. Article (CrossRef Link)

[10] Wan, C. Y., Campbell, A. T., Krishnamurthy L. (2002). PSFQ: A Reliable Transport Protocol for Wireless Sensor Networks. ACM Int'l Workshop on Wireless Sensor Networks and Applications, 1-11. Article (CrossRef Link)

[11] Zhou, H., Wu, Y., Feng, L., Liu, D. (2016). A security mechanism for cluster-based wsn against selective forwarding. Sensors. 16(9), 1537. Article (CrossRef Link)

[12] Khan, W. Z., Xiang, Y., Aalsalem, M. Y. Khan, W. Z., Yang, X., Aalsalem, M. Y., Arshad, Q. (2011). Comprehensive Study of Selective Forwarding Attack in Wireless Sensor Networks. International Journal of Computer Network and Information Security. 3(1), 1-10. Article (CrossRef Link)

[13] Kaplantzis, S., Shilton, A., Mani, N., Sekercioglu, Y. (2007). Detecting selective forwarding attacks in wireless sensor networks using support vector machines. IEEE Int'l Conf. Intelligent Sensors, Sensor Networks and Information, 335-340. Article (CrossRef Link)

[14] Sun, H. M., Chen, C. M., Hsiao, Y. C. (2007). An efficient countermeasure to the selective forwarding attack in wireless sensor networks. TENCON 2007-2007 IEEE Region 10 Conference (pp. 1-4). IEEE. Article (CrossRef Link)

[15] Sun, H. M., Hsu, S. P., Chen, C. M. (2007). Mobile jamming attack and its countermeasures in wireless sensor networks. 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07), 1, 457-462. Article (CrossRef Link)

[16] Krontiris, I., Dimitriou, T., Freiling, F. C. (2007). Toward intrusion detection in wireless sensor networks. Proceedings of the 13th European Wireless Conference (pp. 1-4). Article (CrossRef Link)

[17] Lee, H. Y., Cho, T. H. (2007). Fuzzy-based reliable data delivery for countering selective forwarding in sensor networks. Ubiquitous Intelligence and Computing, 535-544. Article (CrossRef Link)

[18] Hai, T. H., Huh, E.-N. (2008). Detecting Selective Forwarding Attacks in Wireless Sensor Networks Using Two-hops Neighbor Knowledge. IEEE Int'l Symp. Network Computing and Applications. 325-331. Article (CrossRef Link)

[19] Brown, J., Du, X. (2008). Detection of selective forwarding attacks in heterogeneous sensor networks. IEEE Int'l Conf. Communications.1583-1587. Article (CrossRef Link)

[20] Lei, X., Yong-jun, H., Yong, P., Yue-Fei, Z. (2009). A Polynomial based Countermeasure to Selective Forwarding Attacks in Sensor Networks. Int'l Conf. Communications and Mobile Computing. 455- 459. Article (CrossRef Link)

[21] Xin-sheng, W., Yong-zhao, Z., Shu-ming, X., Liangmin, W. (2009). Lightweight defense scheme against Selective forwarding attacks in wireless sensor networks. IEEE Int'l Conf. Cyber-Enabled Distributed Computing and Knowledge Discovery. 226-232. Article (CrossRef Link)

[22] Li, G., Liu, X., Wang, C. (2010). A Sequential Mesh Test based Selective Forwarding Attack Detection Scheme in Wireless Sensor Networks. Int'l Conf. Networking, Sensing and Control. 554-558. Article (CrossRef Link)

[23] Park, J., Seong, D., Yeo, M., Lee, B., Yoo, J. (2013). An Energy-Efficient Selective Forwarding Attack Detection Scheme Using Lazy Detection in Wireless Sensor Networks. In Ubiquitous Information Technologies and Applications. 157-164. Article (CrossRef Link)

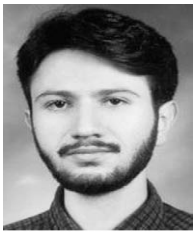
[24] Hu, Y., Wu, Y.M., Wang, H.S. (2014). Detection of Insider Selective Forwarding Attack Based on Monitor Node and Trust Mechanism in WSN. Wirel. Sens. Netw. 6, 237–248. Article (CrossRef Link)

[25] Cu,i B., Yang, S. J. (2014). NRE: Suppress Selective Forwarding Attacks in Wireless Sensor Networks. IEEE Conference on Communications and Network Security (CNS), San Francisco, CA, USA, 229–237. Article (CrossRef Link)

[26] Liao, H., Ding, S. (2015). Mixed and Continuous

Strategy Monitor-Forward Game Based Selective Forwarding Solution in WSN. Int. J. Distrib. Sens. Netw. 11(11), 1-13. Article (CrossRef Link)
[27] Liu, A., Mianxiong, D., Kaoru, O., Jun, L. (2015). PHACK: An Efficient Scheme for Selective Forwarding Attack Detection in WSNs. Sensors. 15(12), 30942-30963. Article (CrossRef Link)
[28] Zhou, H., Yuanming, W., Li, F., Daolei, L. (2016). A Security Mechanism for Cluster-Based WSN against Selective Forwarding. Sensors. 16(9), 1537-1547. Article (CrossRef Link)
[29] Lee, S. B., Choi, Y. H. (2006). A Resilient Packet-Forwarding Scheme against Maliciously Packet-Dropping Nodes in Sensor Networks. ACM Workshop on Security of Ad hoc and Sensor Networks. 59-70. Article (CrossRef Link)
[30] Zhu, S., Setia, S., Jajodia, S. (2003). LEAP, Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks. ACM Int'l Conf. Computer and Communications Security. 752-759. Article (CrossRef Link)
[31] Mathur, A., Newe, T., Rao, M. (2016). Defence against black hole and selective forwarding attacks for medical WSNs in the IoT. Sensors. 16(1), 118-128. Article (CrossRef Link)
[32] Alajmi, N., Elleithy, K. (2015). Multi-layer approach for the detection of selective forwarding attacks. Sensors. 15(11), 29332-29345. Article (CrossRef Link)
[33] Esnaashari, M., Meybodi, M. R. (2017). Dynamic irregular cellular learning automata. Journal of Computational Science. 24, 358-370. Article (CrossRef Link)
[34] Jamshidi, M., Sheikh Abooli Poor S., Nasih Qader, N., Esnaashari, M., Meybodi, M. R. (2019). A Lightweight Algorithm against Replica Node Attack in Mobile Wireless Sensor Networks using Learning Agents. IEIE Transactions on Smart Processing & Computing, 8(1), 58-70. Article (CrossRef Link)
[35] Esnaashari, M., Meybodi, M.R. (2015). Irregular cellular learning automata. IEEE transactions on cybernetics. 45(8), 1622-1632. Article (CrossRef Link)
[36] Thathachar, M. A. L., Bhaskar, R. H. (1987). Learning automata with changing number of actions. IEEE Trans. Systems, Man and Cybernetics. 17(6), 1095-1100. Article (CrossRef Link)
[37] Sobeih, A., Hou, J.C., Kung, L.C., et al. (2006). J-Sim: a simulation and emulation environment for wireless sensor networks," IEEE Wireless Communications. 13, (4), 104-119. Article (CrossRef Link)
[38] Egea-Lopez, E., Vales-Alonso, J., Martinez-Sala, A. S., Pavon-Marino, P., & García-Haro, J. (2005, July). Simulation tools for wireless sensor networks. In Summer simulation multiconference, SPECTS (pp. 2-9). Article (CrossRef Link)
[39] Toor, A. S., Jain, A. K. (2017). A survey on wireless network simulators. Bulletin of Electrical Engineering and Informatics, 6(1), 62-69. Article (CrossRef Link)
[40] Jamshidi, M., Esnaashari, M., Darwesh, A. M., Meybodi, M. R. (2020). Using Time-Location Tags and WNs to Defend Against Node Replication Attack in Mobile Wireless Sensor Networks. International Journal of Wireless Information Networks. 27(1), 102-115. Article (CrossRef Link)
[41] Stafrace, S. K., Antonopoulos, N. (2010). Military tactics in agent-based sinkhole attack detection for wireless ad hoc networks. Computer Communications. 33(5), 619-638. Article (CrossRef Link)
[42] Cao, N., Higgs, R., O'Hare, G. M. (2015). Intelligent Evaluation Models Based on Different Routing Protocols in Wireless Sensor Networks. In Ubiquitous Computing Application and Wireless Sensor, Springer, Dordrecht. 197-209. Article (CrossRef Link)
[43] Jamshidi, M., Esnaashari, M., Darwesh, A. M., Meybodi, M. R. (2019). Detecting Sybil nodes in stationary wireless sensor networks using learning automaton and client puzzles. IET Communications, 13(13), 1988-1997. 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.



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.

Shahin Ghasemi received the B.S. degree in Computer Engineering from the Zagros Higher Education Institute, Kermanshah, Iran, in 2014, and M.S. degree in Information Technology from Islamic Azad University, Kermanshah, Iran, in 2018. His research interests include wireless networks, data mining, metaheuristic algorithms, and cloud computing.

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.

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 from1985 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.