IET Communications

Research Article

IET Journals
The Institution of Engineering and Technology

Detecting Sybil nodes in stationary wireless sensor networks using learning automaton and client puzzles

ISSN 1751-8628
Received on 19th October 2018
Revised 25th April 2019
Accepted on 13th May 2019
doi: 10.1049/iet-com.2018.6036
www.ietdl.org

Mojtaba Jamshidi[1] ✉, Mehdi Esnaashari[2], Aso Mohammad Darwesh[3], Mohammad Reza Meybodi[4]

[1]*Faculty of Computer and Information Technology Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran*
[2]*Faculty of Computer Engineering, K. N. Toosi University of Technology, Tehran, Iran*
[3]*Department of Information Technology, University of Human Development, Sulaymaniyah, Iraq*
[4]*Computer Engineering and Information Technology Department, Amirkabir University of Technology, Tehran, Iran*
✉ *E-mail: jamshidi.mojtaba@gmail.com*

Abstract: A well-known harmful attack against wireless sensor networks (WSNs) is the Sybil attack. In a Sybil attack, WSN is destabilised by a malicious node which forges a large number of fake identities to disrupt network protocols such as routing, data aggregation, and fair resource allocation. In this study, the authors suggest a new algorithm based on a composition of learning automaton (LA) model and client puzzles theory to identify Sybil nodes in stationary WSNs. In the proposed algorithm, each node sends puzzles to its neighbours periodically during the network lifetime and tries to identify Sybil nodes among them, considering their response time (puzzle solving time). In this algorithm, each node equipped with a LA to reduce the communication and computation overhead of sending and solving puzzles. The proposed algorithm has been simulated using J-SIM simulator and simulation results have shown that the proposed algorithm can detect 100% of Sybil nodes and the false detection rate is about 5% on average. Also, the performance of the proposed algorithm has been compared to a wellknown neighbour-based algorithm through experiments and the results have shown that the proposed algorithm is significantly better than this algorithm in terms of detection and false detection rates.

1 Introduction

Wireless sensor networks (WSNs) provide an ideal solution for a variety of monitoring and surveillance applications, including battlefield surveillance, environmental monitoring, and so on. Typical WSNs consist of hundreds or even thousands of sensor nodes deployed over a wide and unattended area. Considering sensor nodes' limitations in terms of memory capacity, energy, transmission range and computational power, and wireless transmission nature and unattended deployment (in many cases), establishing security in WSNs is an important and challenging issue [1].

One of the most famous attacks affecting network layers is a Sybil attack [2]. In a Sybil attack, as it shows in Fig. 1, an adversary injects a malicious node to the network, which begins to exhibit multiple identities ($S1$–$S10$ in Fig. 1), referred to as Sybil nodes hereafter, where they could be either created by the adversary or stolen from normal nodes in the network. By showing multiple identities, the malicious node can mislead genuine nodes to believe that they have many neighbours. Thus it attracts much traffic to itself, disrupting routing protocols and affecting network operations like data aggregation, voting, reputation evaluation, and fair resource allocation [2, 3].

The main contributions of this paper are given as follows:

- Proposing a fully localised algorithm for detecting malicious Sybil nodes in stationary WSNs.
- It is providing a scalable puzzle-based scheme for detecting Sybil nodes. Existing puzzle-based algorithms rely on a central handler for propagating puzzles and validating responses. However, in the proposed scheme, this is done by every node within the network in its neighbourhood. This is better scaled to the area of WSNs.
- Utilising learning automata to reduce the communication and computation overhead of sending and solving puzzles. Learning automata collaboratively try to identify suspicious as well as non-suspicious regions of the network. This information is then used by nodes to send puzzles only in suspicious regions.

The rest of the paper is organised as follows: Section 2 discusses related works. The client puzzles and learning automata are presented in Sections 3. In Section 4, we present the system assumptions and attack model. Section 5 presents the proposed algorithm, while Section 6 presents the simulation results. Finally, the paper is concluded in Section 7.

2 Related work

2.1 Algorithms for stationary WSNs

Newsome *et al.* [3] analysed Sybil attack systematically and presented its taxonomy according to the way Sybil identities are created and simultaneity of indicating them. They also proposed four methods to defend against Sybil attacks, such as radio resource test (RRT), code attestation (CA), random key pre-distribution (RKP), identity registration (IR), and position verification (PV). An RSSI-based algorithm is proposed by Misra and Myneni [4], which can detect Sybil attack even if Sybil nodes change their transmission power for each Sybil node [4]. Jamshidi *et al.* [5] proposed an algorithm which employing some mobile observer nodes to detect Sybil nodes in stationary WSNs. In this algorithm, observer nodes first walk in the network to discover suspicious areas and keep a list of nodes located in such areas in a matrix called MSR in their memory. Then each observer node filters its MSR individually and detects Sybil nodes.

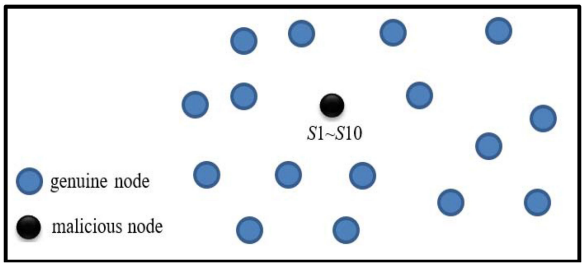


Fig. 1 *Example of Sybil attack*

Ssu *et al.* [6] proposed a distributed algorithm for detecting Sybil nodes, in which node identities are verified by analysing information of each node's neighbouring node. This algorithm is based on the assumption that the probability of two nodes having the same set of neighbours is extremely low.

Rupinder *et al.* [7] proposed a mechanism based on evaluating trust values of neighbour nodes to against Sybil attack in WSNs. The nodes with the trust values less than a threshold value are detected as Sybil nodes. Dhamodharan and Vayanaperumal [8] proposed a message authentication algorithm for detecting Sybil nodes in WSNs. This algorithm uses message authentication and passing procedure for authentication before communication. If a sensor node is not authorised by the base station, it cannot communicate with other nodes in the network.

Rafeh and Khodadad [9] proposed a distributed algorithm based on propagating two-hop messages to detect Sybil nodes in WSNs. In this algorithm, each node discovers its two-hop neighbours and the common neighbours between itself and each of its two-hop neighbours by propagating two-hop messages. The number of common neighbours is used to detect Sybil nodes.

A rule-based anomaly detection system is proposed by Sarigiannidis *et al.* [10]. This expert system relies on an ultra-wideband (UWB) ranging-based detection algorithm that operates in a distributed manner requiring no cooperation or information sharing between the nodes. Tang and Wang [11] proposed a positioning algorithm based on number allocating and mutual guarantee relying on neighbours to detect Sybil nodes in WSNs. The algorithm employs a declaration number and the guaranteed node to mark Sybil nodes in the network. This algorithm also adopts the authentication mechanism of the one-way hash function.

2.2 Algorithms for cluster-based WSNs

Two algorithms were proposed by Chen *et al.* [12] and Jangra and Priyanka [13] to defend against Sybil attack in cluster-based WSNs. These algorithms use RSSI to detect Sybil nodes, which appear as cluster heads. Jan *et al.* [14] proposed a detection algorithm to defend against Sybil attack in centralised clustering-based WSNs. This algorithm uses nodes' cooperation to analyse RSSI of neighbouring nodes. The algorithm executes before the cluster formation to prevent Sybil forged identities from participating in cluster-head selection.

Jamshidi *et al.* [15] proposed a novel model of Sybil attack in cluster-based sensor networks. In this model of Sybil attack, the malicious node uses each of its Sybil identity to join each cluster in the network. Thus, the malicious node joins many clusters of the network simultaneously. They also proposed a distributed algorithm based on RSSI and positioning using three points to defend against this novel attack model.

2.3 Algorithms for mobile WSNs

Piro *et al.* [16] proposed an algorithm to detect Sybil nodes in mobile networks. This algorithm employs some observer nodes, which passively monitor other nodes' traffic in the network and detect Sybil nodes considering that all identities of a single Sybil attacker are bound to a single physical node and must move together. Jamshidi *et al.* [17] also proposed a distributed, lightweight, and dynamic algorithm for detecting Sybil nodes in mobile WSNs. This algorithm uses watchdog nodes first to label (*bit_label*) mobile nodes based on their movement behaviours, and then detects Sybil nodes according to the labels, during the detection phase. Jamshidi *et al.* [18] also proposed another algorithm which uses some observer nodes to detect Sybil attacks in mobile WSNs. This algorithm has two phases: (i) monitoring phase in which observer nodes record the number of meeting occurrences of other nodes in a vector called *history*, for R time periods and (ii) detection phase in which the observer nodes cooperate and identify Sybil nodes based on the content of *history* vectors. Jamshidi *et al.* [19] also proposed a simple and precise algorithm to detect Sybil nodes in mobile WSNs, which avoids observer nodes' cooperation to overcome the drawbacks of their previous algorithm.

3 Background

3.1 Client puzzles

Client puzzle-based approaches are one of the most common techniques used by researchers to protect against network attacks and security problems. Juels and Brainard [20] used client puzzles in TCP SYN flooding; they mentioned the same problem in SSL and gave attentive security proof. Client puzzles also used in authentication protocols by Aura *et al.* [21]. It has also been suggested by Dwork and Naor [22] to regulate measures against junk mail. Singhai and co-authors [23] applied client puzzles to defend against Hello flooding attacks. Client puzzles also used in some algorithms to defend against DOS attacks [24, 25].

To create a puzzle, the server generates a random value Ns of 64 bits of entropy to forbid attacks by estimating the nonce. The value should not be predictable as a time stamp. This entropy should be enough to forbid an attacker to predict 'nonce-result' pairs and the occasional matches caused by birthday attacks would not do too much harm here. It is also the role of the server to determine puzzle difficulty level k, based on the actual conditions. In this case, clients receive the puzzle as a pair: $<< Ns, k >>$, in which, Ns is server nonce and k is puzzle difficulty level [24, 25].

Clients have to generate a nonce Nc before solving a puzzle. This nonce is important on one side to create a new instance by generating a new Nc if the client reuses a server nonce. On the other side, to prevent attackers solve the puzzle and send back the result to the server. Only 24 bits of entropy can be enough to prevent the attacker. To go through the values of Nc since Ns changes regularly. Besides, the client has frequently to run the hash function to a quantity, based on (1), and when first k bits of Y is equal to 0, the puzzle is considered to be solved [24]

$$h(id, Ns, Nc, X) = Y \tag{1}$$

where id is the client identity, h is a cryptographic hash function, such as MD5 or SHA-1 and X is a puzzle solution. Because the server changes Ns periodically, to prevent reusing prior correct solutions, solved instances are kept as a list in the form of (Ns–Nc) pairs.

Brute-force must be used to find X, because there is no known way to discover it. The difficulty level k indicates the time needed to solve a puzzle. If $k = 0$, it means no work is required, but when $k = 128$ (for MD5 function) or 192 (for SHA function), the client has to reverse an entire one-way function which is computationally unbelievable [24, 25].

3.2 Learning automata

A learning automaton (LA) is an abstract model of finite state machine capable of doing finite actions. Actions are evaluated by a stochastic environment, and responses send to the automaton as a positive or negative signal. Then the LA updates its action probability vector based on the received signal. The LA step-by-step learns how to select the optimal solution by repeating this process, and then induce favourable responses from the environment [26, 27].

There are two classes of learning automata: fixed structure learning automata and variable-structure learning automata (VSLA). A variable-structure LA is defined by the quadruple $\langle \alpha, \beta, p, T \rangle$, where α appears for the action set, β appears for the input set, $p = (p_1, p_2, \ldots, p_r)$ appears for the action probability set, and $p(n+1) = T(\alpha(n) + \beta(n) + p(n))$ appears for the learning algorithm. At the time instant n, the LA, according to the action probability set p, selects an action, randomly, and performs it on the environment. The LA updates its action probability set by (2), when it receives the environment's reinforcement signal $\beta(n)$, for favourable responses and by (3) for unfavourable ones [26, 27]

$$\begin{aligned} p_i(n+1) &= p_i(n) + a[1 - p_i(n)] \\ p_j(n+1) &= (1-a)p_j(n) \quad \forall j,\ j \neq i \end{aligned} \tag{2}$$

$$p_i(n+1) = (1-b)p_i(n)$$
$$p_j(n+1) = \frac{b}{r-1} + (1-b)p_j(n) \quad \forall j,\ j \neq i \tag{3}$$

In both above equations, a is the reward and b is the penalty parameter.

4 System assumptions, attack model, and notations

4.1 System assumptions

WSN comprises N sensor nodes randomly distributed in a two-dimensional area. Nodes are unaware of their locations. They are stationary, in that they do not have any movements. Each sensor node has a unique identity. It is assumed that the nodes communicate with one another via a wireless radio channel and broadcast in an omnidirectional mode. All nodes have the same and fixed radio transmission r. it is assumed that the nodes can generate the puzzles, verify the integrity of a solved puzzle, and adjust their difficulty level. It is also assumed that the network is deployed in an adversary environment and so, the nodes might be captured by an attacker. The nodes are not tamper-resistant, so if they are compromised by the adversary, their confidential information will be revealed and the adversary can reprogram them and then send them back to the network as malicious nodes. It is also assumed that nodes use the security scheme, given in [28], to guard the hello messages and puzzles against being spoofed and fabricated by the adversary. This scheme employed secure and lightweight user authentication and key agreement for distributed WSN.

4.2 Attack model

In a Sybil attack, there is a malicious node which exhibits multiple identities, called Sybil nodes. According to a classification proposed by Newsome *et al.* [3], we can classify Sybil attacks in the three orthogonal dimensions:

- *Communication nature:* An attacker can use direct or indirect communication for establishing a Sybil attack. In the direct communication class, Sybil nodes can communicate directly with legal nodes but in the indirect communication class, the malicious node acts as an intermediate node to exchanges packets between legal nodes and the Sybil nodes (no legal nodes can communicate directly with the Sybil nodes).
- *Identity nature:* The Sybil nodes' identity which exhibited by a malicious node, can be fabricated (some new identities) or stolen (from some legal nodes in the network).
- *Simultaneity:* In this dimension, Sybil attacks are classified in simultaneous or non-simultaneous. In the simultaneous class, the malicious node exhibits all its Sybil identities in the network at once. Alternately, in the non-simultaneous class, the malicious node might exhibit a few number of its Sybil identities over a period of time.

In this work, like [6, 7, 9, 10, 15, 17–19], we only consider the direct and simultaneous Sybil attacks with both stolen and fabricated identifiers. Furthermore, as stated in [6], in the attack model, it is assumed that the network is insecure and that the nodes could be captured with a certain probability. Also, our attack model assumes that the adversary does not employ external powerful devices to lunch Sybil attacks, as it is assumed in many existing papers [6, 9, 10, 11, 15, 17–19]. Like [6, 9, 10, 11, 15, 17–19], it is assumed that M ($M < N$) legitimate nodes of the network can be captured by an adversary and reprogrammed to establish a Sybil attack. Hence, we assume the equality of capability (regarding computational, memory and etc.) between malicious nodes and benign ones. According to this model, injecting some outside-powerful malicious nodes by the adversary is impossible. While an adversary can completely take over nodes, we assume that such an adversary cannot outnumber legitimate nodes by replicating captured ones or introducing new ones in sufficiently many parts of the network [10].

Basically, nodes are considered to be benign except compromised ones which are considered as malicious nodes. These malicious nodes create multiple fake identities (Sybil nodes) and try to cheat their neighbours. Sybil nodes also reply to messages received from nodes (if needed). It is assumed by the attack model that the malicious nodes will forge a new identity for each entity which is created. The main mission of the malicious node is to trick benign nodes in the network into believing that they have many neighbours. Since these nodes do not, in fact, exist, many of the network protocols are seriously disturbed or rendered inoperable [6, 10]. And the last assumption, according to [9], is that Sybil nodes do not execute the detection algorithm to save their energy and to prevent network benign nodes from becoming suspicious of their presence.

4.3 Notations

In this paper, the following notations are used:

- Ψ is the set of all nodes, including normal and Sybil nodes.
- ξ is the set of all normal nodes.
- Ns, Nc and k are server nonce, client nonce and difficulty level of the puzzles, respectively.
- R_i represents the ith round of propagating the puzzles in the network.
- $\mathfrak{R}$ represents the number of rounds in which the proposed algorithm should be executed.
- LA_v represents v nodes' LA.
- p represents action probability set for a LA.
- $Action$ and old_{Action} represent the selected action by a LA in current and previous rounds, respectively.
- $Result$ and old_{Result} represent the result of solving puzzles that turn back to a node from its neighbours, in current and previous rounds, respectively.
- NB_v represents the set of v nodes' neighbours, $v \in \psi$.
- $|NB_{AVG}|$ represents the number of node's neighbours, on average case.
- $RNB_v^+ \subseteq NB_v$ represents the set of those v nodes' neighbours that acquire positive score because of solving the puzzle, $RNB_v^+ \subseteq NB_v$.
- RNB_v^- represents the set of those v nodes' neighbours that acquire negative score because of solving the puzzle wrongly or with a high delay in solving it, $RNB_v^- \subseteq NB_v$.
- T_m is a threshold to specify when a LA gift reward or penalty is assigned to the current selected action.
- $Response_{map}$ is a bitmap matrix that a node uses to keep the received old_{Result} from its neighbours temporarily.

5 Proposed algorithm

The main idea of the proposed algorithm is adapted from this fact that all queries and send packets from normal nodes to Sybil nodes are processed only by a single processor unit and transceiver through a channel because all Sybil nodes are on the same hardware (the malicious node). So by considering this assumption, the capability of malicious nodes and normal nodes are equal, the response time for Sybil nodes, in the situation that is required for all Sybil nodes to perform an operation and respond simultaneously, is much higher than normal nodes.

In the proposed algorithm, we use this subject to detect Sybil nodes, such that normal nodes propagate the puzzles to neighbours and wait for their response (solving puzzles) during the network's lifetime. Since the Sybil nodes related to a malicious node respond with high latency, so the Sybil nodes can be simply distinguished from normal nodes. This means that the proposed algorithm is iterative in nature, so multiple rounds are executed during the network's lifetime. Also, with respect to high communication and computation overhead of sending and solving the puzzles, we equip each sensor node with a LA to decrease these overheads, by reducing the number of puzzles that each node should propagate to

Neigbor_ID	Score
u	0
v	0
w	0

Fig. 2 *Structure of neighbourhood table for sensor nodes*

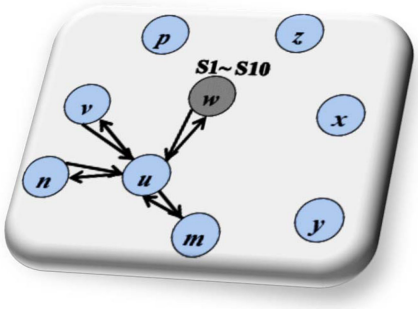


Fig. 3 *View of the proposed algorithm*

its neighbours, during the network's lifetime. We describe the proposed algorithm in details in the following.

In the proposed algorithm, each sensor node v have a neighbourhood table that is shown in Fig. 2, where it stores identities of own neighbours in *Neighbor_ID* column and keeps the donative scores to each of its neighbours in *the Score* column. After nodes deployment phase, each node broadcasts a hello message to show itself to the neighbour nodes. Therefore, each node discovers its own neighbours and stores their identities in its neighbourhood table. In this step, the *Score* field of neighbourhood table is set by 0, for all neighbours.

Each node is also equipped with a LA, with two possible actions, $\alpha = \{\alpha_0, \alpha_1\}$. Action α_0 represents not propagating the puzzles and action α_1 represents the propagating the puzzles. In fact, for a node, action α_0 indicates that puzzle propagation is unnecessary because probably there exist no Sybil nodes in its neighbourhood, and action α_1 indicates that puzzle propagation is necessary because probably there exist Sybil nodes in its neighbourhood and so it should send the puzzles to its neighbours in order to distinguish the Sybil neighbours from normal ones, according to the responses (solving puzzles) returned from them. In the beginning, the probability value for these two actions, according to (4), is equal to 0.5.

$$p = \{\overbrace{0.5}^{\alpha_0}, \overbrace{0.5}^{\alpha_1}\} \tag{4}$$

As mentioned above, the proposed algorithm should be executed in multiple rounds, R_i, during the network's lifetime. At each R_i, all nodes' learning automata are activated simultaneously based on the action probability set p, and action is chosen randomly. As presented in Fig. 3, each node v generates a puzzle, propagates it to its neighbours and then waits for their response for Δ time, if LA_v chooses an action α_1, neighbours of node v, such as node u, upon receipt of this puzzle, should solve the puzzle upon receipt, and sends the result back to node v. Now, if there is a malicious node in v's neighbourhood, it must solve the puzzle and send the result back to node v, for each of its own Sybil identities. It is clear that the malicious node (w in Fig. 3) responds very late, for some of its Sybil identities. So node v gives them a negative score.

Node v stores the response time of each neighbour in the *Result* vector. At the beginning of the next round, R_{i+1}, each node sends its *Result* vector (related to R_i) along with the puzzle (related to R_{i+1}) to its neighbours, if it has propagated the puzzle at R_i (its LA has chosen action α_1), so at R_{i+1}, each node will have a *Result* vector of its neighbour at R_i and stores them in its own $Response_{map}$ matrix, for the nodes to discover their neighbours nature precisely. For example, a specific node, such as u, it is possible that it receives multiple puzzles from some neighbours (like w, v) at R_i. In this situation, node u solves and sends the solution to the puzzle of node w back timely but responds to node v with more delay. But, when node v receives R_i's *Result* vector from node w at R_{i+1}, it is to be aware that node u have also solved another puzzle (the puzzle of node w) at R_i, so node v allows the node u's delay.

This process, sending the *Result* vector to the neighbours, leads to higher accuracy of the proposed algorithm in discovering the nature of the nodes. So in general, node v, at R_{i+1}, regarding its own R_i's *Result* vector and the others received from its neighbours, increments/decrements *Score* field of its neighbours (in neighbourhood table) by 1 unit. For each node v, if ratio of the number of neighbours that have a negative score to all of the neighbours is greater than threshold T_m, LA_v rewards action α_1 and updates its action probability set based on (2), otherwise, it penalises this action and updates its action probability set based on (3).

Thus, by increasing the number of rounds of the proposed algorithm, the nodes' learning automata learn whether there exists a malicious node (or nodes) in their neighbourhood or not. The nodes propagate the puzzles to their neighbours if there exists a malicious node in their neighbourhood. Otherwise, they do not propagate the puzzles, and this decreases communication and computation overheads, and hence decreases energy consumption.

The time interval between variant rounds of execution of the proposed algorithm, T_{Round} is calculated according to the following equations:

$$\begin{aligned} T_{\text{Round}} &\geq |NB_{\text{AVG}}| \times \Delta \\ \Delta &= \text{Time}(k) + T_d + P_d \\ T_d &= \frac{\text{packet size (bit)}}{\text{bandwidth}},\ P_d = \frac{\text{distance}}{\text{speed in medium}} \end{aligned} \tag{5}$$

where Time(k) is the time required to solve a puzzle with difficulty level k, T_d is transmission delay and P_d is propagation delay.

The proposed algorithm is executed $\mathfrak{R}$ rounds and after $R_{\mathfrak{R}}$, each node only trusts those neighbour nodes that have acquired positive scores, and communicate with them. Also, those neighbours that have acquired a negative score are marked as Sybil nodes. The pseudocode of the proposed algorithm is demonstrated in Algorithm 1.

In step 1, for all nodes, the *Result* vector and *Action* field should be initialised with *NULL* and α_0, respectively. Because this is the first round and no puzzle is propagated already.

In step 2, the phase of sending puzzles will be started. In this step, at first, the previous chosen action is assigned to old_{Action} and then the learning automata choose a new action. Now, for each node, if the new action or previous action is α_1, the algorithm enters the puzzles sending process. Otherwise, the node does not send any puzzles to their neighbours. Thus, each node v propagates puzzle to its neighbours at R_i, if any of the following conditions are satisfied:

1. In the condition that $Action = \alpha_1 and old_{Action} = \alpha_0$, node v sends a packet containing a puzzle with difficulty level k and old_{Result} vector (which is NULL) to its neighbours. Then node v waits for its neighbour's responses and registers the results in its *Result* vector.
2. In the condition that $Action = \alpha_0 and old_{Action} = \alpha_1$, node v does not require sending the puzzle. But, since this node sent a puzzle in the previous round, it must send the own old_{Result} to its neighbours. Therefore, it sends a packet containing a puzzle with difficulty level 0 (that not required to be solved) and the old_{Result} vector. Each neighbour of node v stores node v's old_{Result} in its own $Response_{map}$ matrix upon receiving this packet, if action α_1 is chosen at the previous round (means, their $old_{Action} = \alpha_1$). Also, these neighbour nodes neither solve node v's puzzle nor respond to v.

3. In the condition that $Action = \alpha_1 and old_{Action} = \alpha_1$, node v sends a packet containing a puzzle with difficulty level k and the old_{Result} vector (which is not NULL) to its neighbours. Then, node v waits for neighbour responses and registers the results in its *Result* vector. Each neighbour of node v stores the node v's old_{Result} in its own $Response_{map}$ matrix upon receiving this packet, if $old_{Action} = \alpha_1$. In addition, these neighbours must solve node v's puzzle and respond to it as soon as possible.

Step 3 will be executed only by those nodes. With $old_{Action} = \alpha_1$. These nodes first calculate RNB_v^+ and RNB_v^- based on their own old_{Result} and $Response_{map}$. Then, their learning automata are updated based on the ratio of RNB_v^- to NB_v.

Steps 2 and 3 are repeated $\mathfrak{R}$ rounds and then each node can distinguish the Sybil nodes (if exist Sybil nodes in its neighbourhood) from normal nodes, based on their *Score* field in its neighbourhood table. The nodes with a negative score are marked as Sybil nodes.

Upon the detection of probable Sybil nodes, a node v generates a message including a list of detected Sybil nodes and transmits it to its neighbours. Neighbouring nodes will not communicate with these Sybil nodes anymore. Moreover, node v transmits a copy of this message to the base station so that the network manager is informed that the network has been attacked to undertake stricter security actions (Fig. 4).

6 Performance evaluation and simulation results

In this section, we first evaluate the overhead of the proposed algorithm in terms of memory, communication, and computation. Then, we simulate the proposed algorithm and evaluate its detection rate through some experiments. We also compared its detection rate with the other existing algorithms.

6.1 Performance evaluation

Memory overhead: In the proposed algorithm, each node requires a $2 \times |NB_{AVG}|$ space to store the neighbours' identities and their scores (neighbourhood table), a $2 \times |NB_{AVG}|$ space to keep the neighbours responses at current and previous rounds (*Result* and old_{Result} vectors), and a $|NB_{AVG}| \times |NB_{AVG}|$ bit space to store $Response_{map}$ (its neighbours' old_{Result}). Therefore, the memory overhead of the proposed algorithm is $O(|NB_{AVG}|^2)$ bits.

Communication overhead: Energy consumption of an algorithm is critical due to the limitation of sensor nodes' energy. Since sending packets consumes far much energy in comparison to other operations such as receiving or computing. Therefore the number of transmitted packets imposed upon the network during execution of a certain algorithm is considered as a significant criterion. At each round of the proposed algorithm, each node sends a packet containing a puzzle and old_{Result} to its neighbours and also responds to them (sends back the solutions of their puzzles). So, in the worst case, each node must send $|NB_{AVG}| + 1$ packets, in each

Algorithm 1. Pseudo code of the proposed algorithm

1: for each $v \in \xi$ Do $Action = \alpha_0$. result = NULL
2: for each $v \in \xi$ Do
  $old_{Action} = Action$
  $Action = LA_v$ select an Action from its action set $\{\alpha_0.\alpha_1\}$
  if $(Action = \alpha_1$ or $old_{Action} = \alpha_1)$
    $old_{Result} = Result$
    if$(Action = \alpha_1)$
      create a puzzle $P < N_s.K >$
    else
      create a puzzle $P < N_s.0 >$
    end if
    node v broadcast a packet.containing a puzzle P and its old_{Result}.to its neighbors
    for each node $u \in NB_v$ that it receives v's packet Do
      if $(old_{Action} = \alpha_1)$
        register old_{Result} received from v in its $Response_{map}$
      end if
      if $(K > 0)$
        solving puzzle P so that $h(u.N_S.N_C.X) = Y$
        send back the solved puzzle to node v
      end if
    end for
    node v. upon receiving the solved puzzles.verifies and register them into its Result vector
  end if
 end for
3: for each $v \in \xi$ Do
  if $(old_{Action} = \alpha_1)$
    update score of its neighbors
    computes RNB_v^+ and RNB_v^- .according to its old_{Result} and Response_map
    if (ratio of RNB_v^- to NB_v is greater than T_m)
      LA_v gift reward to Action (α_1)
    else
      LA_v penalty Action (α_1)
    end if
  end if
 end for
4: repeat steps 2.3 for $\mathfrak{R}$ rounds in the time period of T_{Round}
5: for each $v \in \xi$ Do
  for each $u \in$ neighborhood table Do
    if $(score < 0)$
      mark u as Sybil node
    end if
  end for
 end for

Fig. 4 *Algorithm 1: Pseudo code of the proposed algorithm*

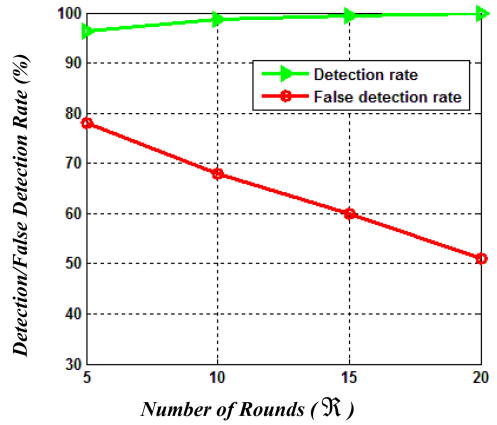


Fig. 5 *Detection rate and false detection rate of the proposed algorithm for* $N = 300, T_m = 0.7, S = 10, a = b = 0.05$ *and* $\mathfrak{R} = 5 - 20$

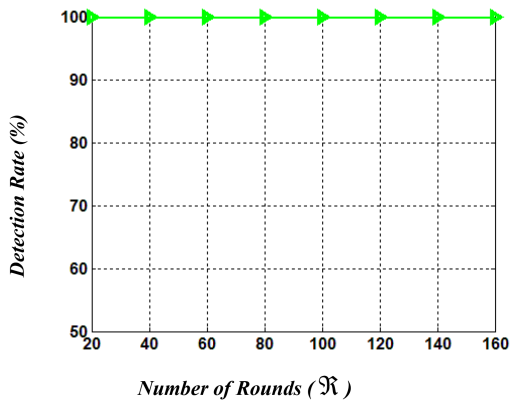


Fig. 6 *Detection rate of the proposed algorithm for* $N = 300, T_m = 0.7, S = 10, a = b = 0.05$ *and* $\mathfrak{R} = 20 - 160$

round. However, by increasing the rounds, the learning automata learn which regions embrace malicious nodes. So puzzles are only sent to the nodes that are located in those regions. By assuming that each node has $|NB_{AVG}|$ neighbours, and there are M malicious nodes in the network, the total sending packets at each round is almost $M \times (|NB_{AVG}| + 1)$. The experiment results presented in the next subsection also show that by increasing the rounds, the communication overhead decreases significantly.

Computation overhead: Considerable computation overhead of the proposed algorithm is due to the time of solving puzzles. Each node must solve $|NB_{AVG}|$ puzzles at each round, in the worst case. But, as mentioned earlier, by increasing the rounds, the learning automata learn which regions embrace malicious nodes, and thus, puzzles are only sent to the nodes that are placed in such regions. This decreases the computation overhead of the proposed algorithm significantly.

To complete the computational overhead of the proposed algorithm, we have to consider the overhead of solving a single puzzle of level k. A puzzle of level k is a partial hash for which k bits are left out and the solver is asked to find out those k bits. To this end, the solver has to guess different strings and try them until it could be able to find out the bits. According to [24, 25], the time required to solve the puzzle with difficulty level k in an *np* problem. To solve a puzzle with difficulty level k, the client needs to perform an average of 2^k–1 operations. Thus, for example, if $k = 20$, then about 1 million operations are needed for solving the puzzle.

Dutta *et al.* [29] presented a detailed performance analysis for cryptography primitives in TinyOS environment. They adapted RSA-1024 and SHA-1 to the Telos. The primitive operation for RSA-1024 was a modular exponentiation with $e = 3$ and for SHA-1 were computing a 160-bit ($k = 160$) hash with 64-byte blocks over 64 bytes. The results of the experiments in [29] were as follows:

(i) Executing the RSA-1024 takes 0.7 s and consumes 529 bytes RAM and 1.2 KB ROM.
(ii) Executing the SHA-1 takes 0.014 s and consumes 95 bytes RAM and 2.3 KB ROM.

However, according to the capabilities of the nodes used in the network, network managers can adjust the puzzles' difficulty level, k, such that solving them does not impose a high processing overhead on sensor nodes.

6.2 Simulation results

The performance of the detection algorithm was evaluated by performing a series of experiments. Three performance metrics were considered here:

- *Detection rate:* percentage of Sybil nodes identified by the security algorithm.
- *False detection rate:* percentage of normal nodes erroneously identified as Sybil nodes.
- *Communication overhead:* the number of sent packets imposed on the network by security algorithm. In the proposed algorithm, this metric is the same send puzzles. Furthermore, the experiment results of the proposed algorithm compared to some other existing algorithms.

The simulation environment was implemented using J-SIM simulator [30]. In simulations, it was assumed that the network is comprised of N sensor nodes, which are randomly distributed within a 100×100 m^2 area. The sensing field contained $M = 10$ malicious nodes where each node forges S Sybil nodes. Both normal and malicious nodes have a fixed communication range equal to $r = 10$ m. We also use puzzles with difficulty level $k = 10$. To ensure the validity of the results, each simulation was repeated 40 times and the results were then averaged to obtain a final value. Experiment results were compared to some other algorithms which are described in Section 2 briefly.

Experiment 1: In this experiment, the effect of parameters $\mathfrak{R}$ and T_m, which are specific to the proposed algorithm, on the detection accuracy of the proposed algorithm is evaluated. In this experiment, we set $N = 300$, $S = 10$ and the reward and penalty parameters of learning automata as $a = 0.05$, $b = 0.05$.

We have first executed the experiment for $T_m = 0.07$ and $\mathfrak{R} = 5 - 20$. The results are shown in Fig. 5 in terms of both detection and false detection rates. These results show that the detection rate of the proposed algorithm for $\mathfrak{R} = 5$ is 96%. However, it yields a high false detection rate; about 78%.

The results show that by increasing $\mathfrak{R}$ from 5 to 20, the detection rate reaches 100% and the false detection rate is reduced to 51%. It is clear that detecting all Sybil nodes in the network takes several rounds because the malicious node solves puzzles timely for some of its own Sybil nodes (like one or two of them). Also, it is possible that the LA of some normal nodes chooses the action α_0, in some rounds and as a result, they do not send the puzzles. Therefore, they are not able to detect the Sybil nodes. For this reason, detecting all Sybil nodes in the network takes several rounds.

The results of this experiment show that although the detection rate of the proposed algorithm for $\mathfrak{R} = 20$ is 100%, the false detection rate is not tolerable. Therefore, the proposed algorithm must be executed for more rounds to achieve a tolerable false detection rate.

We have repeated the experiment for $\mathfrak{R} = 20 - 160$ and obtained results are shown in Figs. 6 and 7 in terms of detection rate and false detection rate, respectively.

Fig. 6 shows that the detection rate of the proposed algorithm for $\mathfrak{R} \geq 20$ is also 100%. In addition, the results in Fig. 7 show that the false detection rate of the proposed algorithm decreases by increasing $\mathfrak{R}$, significantly. The false detection rate is also evaluated for different values of threshold T_m. The results show that by decreasing T_m, the false detection rate of the proposed

algorithm also decreases, slightly. Because the smaller is this threshold, more puzzles are sent by nodes and this reduces false detection rate. However, for $\Re \geq 120$, the false detection rate for different values of T_m, is approximately the same. After 120 rounds, this rate is about 5%, after 140 rounds it is about 4%, and after 160 rounds, it is about 3%. This is because by increasing the number of rounds and sending puzzles, each node can discover the nature of its neighbours precisely.

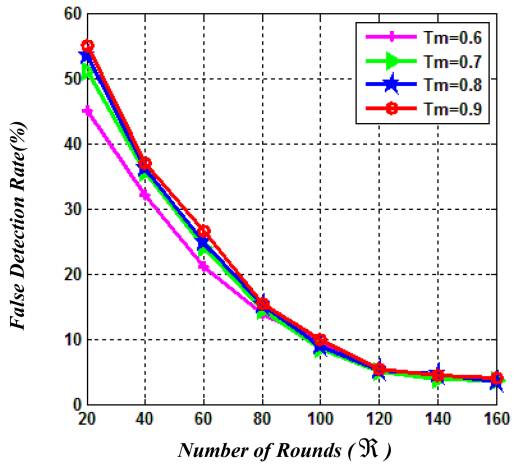


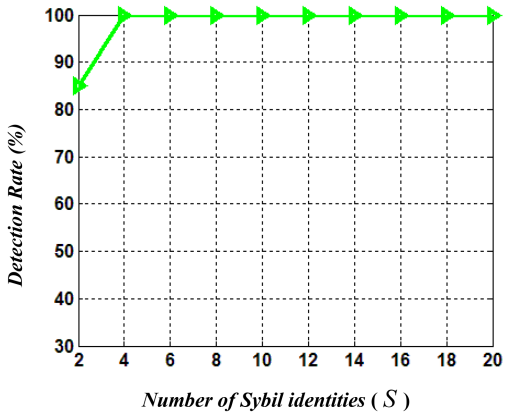
Fig. 7 *Effect of threshold T_m and parameter $\Re$ on false detection rate of the proposed algorithm for $N = 300, a = 0.05, b = 0.05, S = 10$ and $\Re = 20 - 160$*

Experiment 2: In this experiment, we study the effect of parameter S (the number of Sybil identities spread by each malicious node) on both criteria, detection and false detection rates. Here, the parameters $N = 300$, $T_m = 0.7$, $a = 0.05$, $b = 0.05$ are considered. The parameter S varies from 2 to 20 (by the increment step 2). The results of this experiment are obtained after 100 rounds (means, $\Re = 100$). Figs. 8*a* and *b* show the detection rate and false detection rate, respectively.

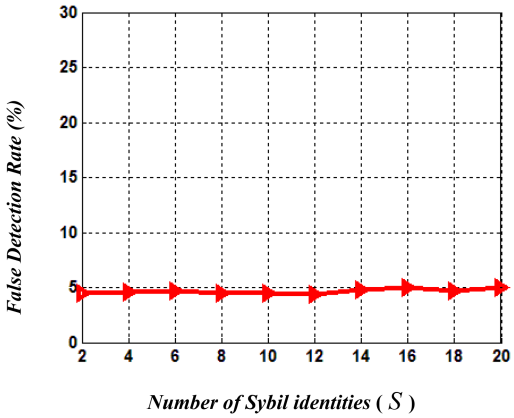
As shown in Fig. 8*a*, the detection rate of the proposed algorithm, in weakest case ($S = 2$) is 85% and for other cases is 100%. When the malicious nodes propagate only two fake identities, they can solve the puzzles for both of their Sybil nodes timely. Therefore, the detection rate of the proposed algorithm is lower than 100%. However, it is clear that in this case, $S = 2$, the malicious nodes cannot disrupt the network operations seriously. Also, the results of this experiment in Fig. 8*b* show that the false detection rate of the proposed algorithm is not influenced by varying the number of Sybil nodes and it is ∼5%.

However, the detection rate of many other algorithms likes [6, 9] is influenced by varying the number of Sybil nodes because their mechanism is relying on the number of Sybil identities which propagates by a malicious node. For example, the algorithm [6] uses the number of nodes' neighbours to detect Sybil nodes. So, when the number of nodes' neighbours exceeds a threshold, the detection mechanism will be executed, i.e. when the number of Sybil nodes is too high. Fig. 9 shows the effect of the number of Sybil nodes on the detection accuracy of the algorithm proposed in [6]. As it is obvious in Fig. 9*b*, the false detection rate of this algorithm is affected by changing the parameter S, significantly.

Experiment 3: Here, the effect of parameter N (total number of nodes) on the performance of the proposed algorithm is evaluated.

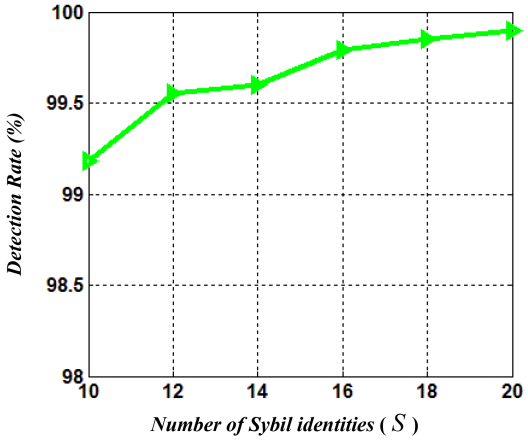


a

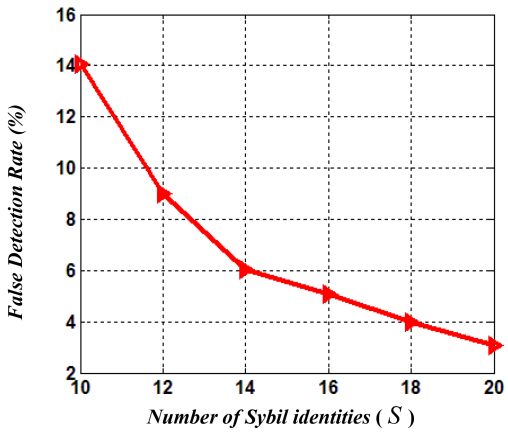


b

Fig. 8 *Effect of number of Sybil nodes, S, on*
(a) Detection rate and, ***(b)*** False detection rate of the proposed algorithm for $N = 300, T_m = 0.7, \Re = 100, a = b = 0.05$ and $\Re = 100$



a



b

Fig. 9 *Effect of the number of Sybil nodes, S, on*
(a) Detection rate and, ***(b)*** False detection rate of the algorithm [6] for $N = 300$

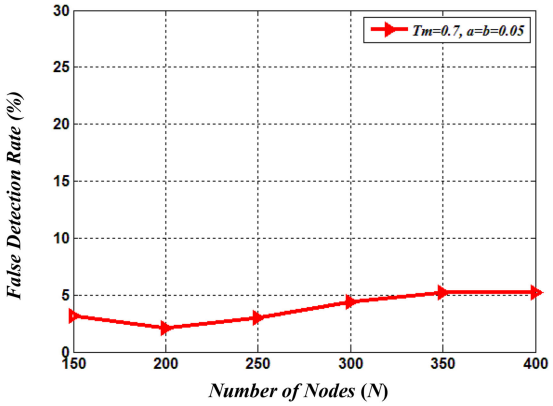


Fig. 10 *Effect of number of nodes, N, on the false detection rate of the proposed algorithm for* $S = 10, T_m = 0.7, a = b = 0.05$ *and* $\Re = 100$

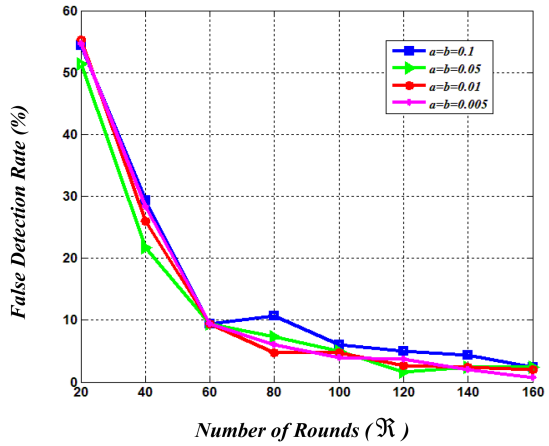


Fig. 11 *Effect of parameters' reward* ***(a)*** and penalty, ***(b)*** On the false detection rate of the proposed algorithm for $N = 300, T_m = 0.7, S = 10$

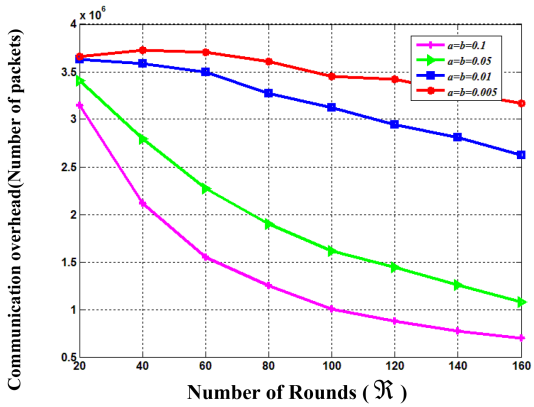


Fig. 12 *Effect of parameters a and b on communication overhead of the proposed algorithm for* $N = 300, T_m = 0.7, S = 10$

In this experiment, parameters $S = 10$, $T_m = 0.7$, $a = 0.05$, $b = 0.05$ are considered. The performance of the proposed algorithm is investigated in 100 rounds. The results of this experiment are evaluated for $N = 150 - 400$. The results show that the number of nodes does not affect true detection and this rate is 100%. Also, Fig. 10 shows the results in terms of the false detection rate. It can be clearly seen that increasing the total number of nodes, increases false detection rate smoothly because by increasing the network density, each node must solve more puzzles. So, some normal nodes may have latency in responding to some of their neighbours. Therefore, the false detection rate increases.

Experiment 4: The goal of this experiment is to evaluate the effect of parameters' reward (a) and penalty (b) on the false detection rate of the proposed algorithm. In this experiment, parameters $N = 300$, $T_m = 0.7$, $S = 10$ are considered and parameters a and b are varied. The results of this experiment are shown in Fig. 11. As it can be seen by decreasing these two parameters, the false detection rate also decreases. Because, when it chooses the lower value for parameters reward and penalty, the learning automata adjust their actions' probability more accurately. This way, some normal nodes which have solved the puzzles with latency, due to being busy, are not marked as Sybil nodes easily. Though, as it can be seen in Fig. 11, after 160 rounds, false detection is reduced to 3% approximately.

Experiment 5: The goal of this experiment is to evaluate the communication overhead of the proposed algorithm. In this experiment, parameters $N = 300$, $S = 10$, $T_m = 0.7$ are considered and we evaluate the effect of parameters reward (a), penalty (b), and $\Re$ of the proposed algorithm. Fig. 12 shows the results of this experiment. As we can see, by increasing the number of rounds, the communication overhead of the proposed algorithm is reduced significantly, because as mentioned before, by increasing the number of rounds, the learning automata learn which regions embrace malicious nodes. So the puzzles are only sent to the nodes that are located in those regions. Thus, the communication overhead decreases significantly. Also, the results show that this metric falls down when a greater value is chosen for reward (a) and penalty (b), because in this case, the learning automata discover the suspicious regions more quickly. So only the nodes that are located in these suspicious regions perform the algorithm (sending the puzzles) and this reduces the communication overhead.

Experiment 6: In this experiment, we evaluate the effect of threshold T_m on the communication overhead of the proposed algorithm. Parameters $N = 300$, $S = 10$, $a = 0.05$, $b = 0.05$ are considered and threshold T_m varies from 0.6 to 0.9 and its influence on the communication overhead is examined. The results in Fig. 13 show that by increasing T_m, the communication overhead is reduced and conversely, by decreasing T_m, communication overhead also increases. The reason is obvious, as noted earlier, at each round of the proposed algorithm, if the number of neighbours with negative scores is greater than threshold T_m, the learning automata rewards action α_1 (sending the puzzles) otherwise, it is penalised. Thus, the greater is this threshold, the action α_1 is penalised more and therefore, fewer puzzles are sent by the nodes and this reduces the communication overhead.

Moreover, the results of the proposed algorithm (for $T_m = 0.7$) in Fig. 14 are compared with the other algorithms. Since the communication overhead of algorithms [6, 9] is very high, their values are divided by 30 to increase the resolution; they are shown in the diagram. In all compared algorithms, the communication overhead at each round of executing the algorithm is constant, but in the proposed algorithm, as the number of rounds increases, learning automata learn suspicious areas and only normal nodes located in suspicious areas execute the Sybil detection algorithm. Therefore, as the number of rounds increases, the communication overhead of the proposed algorithm decreases. Moreover, the communication overhead of the algorithm [17] is less than other algorithms because this algorithm is specifically designed for mobile sensor networks and employs the mobility of nodes to detect Sybil nodes. But this algorithm cannot be employed in stationary sensor networks. Algorithms [4, 12] are also based on RSSI and for iteration rounds less than 140, the communication overhead would be less than other algorithms. In general, RSSI-based algorithms do not perform accurately in noisy environments.

6.3 Further discussions

Adding new nodes to the network: When a new node u is added to a region of the network, it should broadcast a hello message so that neighbouring nodes like node v are informed of its existence and start communicating with the new node. Node v adds the identity

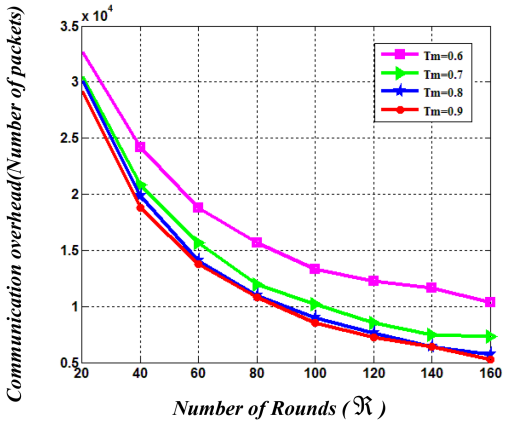


Fig. 13 *Effect of threshold T_m on communication overhead of the proposed algorithm for $N = 300, S = 10, a = b = 0.05$*

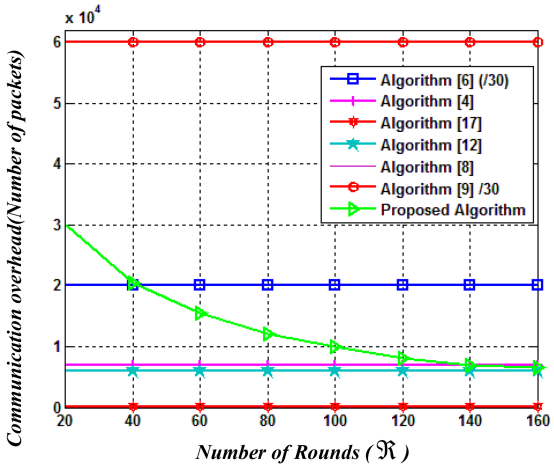


Fig. 14 *Comparison of the proposed algorithm and the other algorithms in terms of the communication overhead for $N = 300, S = 10, a = b = 0.05$*

of u to its neighbourhood table and assigns a zero score to it. Therefore, the execution of the algorithm is not interrupted.

Dynamic link problem: If some links fail temporarily and destination node cannot return the puzzle solution on time, a negative score is assigned to that node and when mentioned link connects, it can obtain a positive score by solving the puzzles again. This prevents it from being selected as a malicious Sybil node by the proposed algorithm. It should be noted that the proposed algorithm does not mark malicious nodes only at one round of transmitting puzzles, but it repeats transmitting puzzles for several rounds ($\Re$times) and then determines Sybil nodes based on the scores.

Various Sybil models: There are many different classes of Sybil attacks reported in the literature. These are direct/indirect communication, fabricated/stolen identities, and simultaneous/non-simultaneous [3]. Among these classes, the proposed algorithm can detect all, but the non-simultaneous class.

Drawbacks: One drawback of the proposed algorithm is that it imposes high communication and computation overhead on legal nodes located in the neighbourhood of Sybil nodes. In future work, we plan to utilise a LA in each node, v, whose duty is to choose neighbouring nodes, which should solve puzzles of v. This new LA learns which neighbouring nodes are legal over time and does not need to solve puzzles. This way, the communication and computation overheads imposed on legal nodes are reduced significantly.

Another drawback of the proposed algorithm is its assumption that an adversary does not employ external powerful devices to lunch Sybil attacks. This limitation exists in many state-of-the-art algorithms like [6, 9, 10, 11, 15, 17–19]. A possible solution for overcoming this limitation is to utilise time-dependent client puzzles which do not permit powerful devices to solve puzzles faster than usual cases.

7 Conclusion

In this paper, we proposed a new algorithm for detecting Sybil nodes in WSNs. Our proposed algorithm is based on the learning automata model and client puzzles theory. In the proposed algorithm, each node sends puzzles to its neighbours periodically, during the network's lifetime, and tries to identify the Sybil nodes among them considering their response time. The learning automata are also used for reducing the communication and computation overhead of sending and solving the puzzles. The proposed algorithm has been simulated and simulation results showed that the proposed algorithm can detect 100% of Sybil nodes after 20 rounds of execution. Also, according to the simulation results, the proposed algorithm exhibited significantly better performance in terms of the false detection rate and the communication overhead when the threshold parameter, T_m, was adjusted to 0.9 and both reward (a) and penalty (b) parameters were set to 0.1. In this condition, the false detection rate for the proposed algorithm was less than 5%, after 100 rounds of execution. To achieve this performance, the maximum communication overhead imposed on each node was $100 \times (|\mathrm{NB}_{\mathrm{AVG}}| + 1)$ packets. The simulation results indicated the proposed algorithm is robust to defend against Sybil attacks. Because it has a high detection rate against Sybil attacks. Furthermore, the performance of the proposed algorithm is compared with some existing algorithms through some experiments and results show that the proposed algorithm outperforms other algorithms in terms of detection rate and false detection rate significantly.

8 References

[1] Jamshidi, M., Shaltooki, A.A., Dagalzadeh, Z., *et al.*: 'A dynamic ID assignment mechanism to defend against node replication attack in static wireless sensor networks', *Int. J. Inf. Vis.*, 2019, **3**, (1), pp. 13–17

[2] Douceur, J.R.: '*The sybil attack, first international workshop on peer-to-peer systems (IPTPS)*' (Springer, Berlin, Heidelberg, 2002), pp. 251–260

[3] Newsome, J., Shi, E., Song, D., *et al.*: 'The sybil attack in sensor networks: analysis and defenses'. Int. Symp. on Information Processing in Sensor Networks, ACM, Berkeley, CA, USA, April 2004, pp. 259–268

[4] Misra, S., Myneni, S.: 'On identifying power control performing sybil nodes in wireless sensor networks using RSSI'. Global Telecommunications Conf., Miami, FL, USA, December 2010, pp. 1–5

[5] Jamshidi, M., Ranjbari, M., Esnaashari, M., *et al.*: 'A new algorithm to defend against sybil attack in static wireless sensor networks using mobile observer sensor nodes', *Ad Hoc Sensor Wirel. Netw.*, 2019, **43**, pp. 213–238

[6] Ssu, K.F., Wang, W.T., Chang, W.C.: 'Detecting sybil attacks in wireless sensor networks using neighboring information', *Comput. Netw.*, 2009, **53**, (18), pp. 3042–3056

[7] Rupinder, S., Singh, J., Singh, R.: 'TBSD: a defend against sybil attack in wireless sensor networks', *Int. J. Comput. Sci. Netw. Secur. (IJCSNS)*, 2016, **16**, (11), pp. 90–99

[8] Dhamodharan, U.S., Vayanaperumal, R.: 'Detecting and preventing sybil attacks in wireless sensor networks using message authentication and passing method', *Scientific World J.*, 2015, **1**, (1), pp. 13–17

[9] Rafeh, R., Khodadadi, M.: 'Detecting sybil nodes in wireless sensor networks using two-hop messages', *Indian J. Sci. Technol.*, 2014, **7**, (9), pp. 1359–1368

[10] Sarigiannidis, P., Karapistoli, E., Economides, A.: 'Detecting sybil attacks in wireless sensor networks using UWB ranging-based information', *Expert Syst. Appl.*, 2015, **42**, (21), pp. 7560–7572

[11] Tang, Q., Wang, J.: 'A secure positioning algorithm against sybil attack in wireless sensor networks based on number allocating'. 17th Int. Conf. on Communication Technology (ICCT), Chengdu, China, October 2017, pp. 932–936

[12] Chen, S., Yang, G., Chen, S.: 'A security routing mechanism against sybil attack for wireless sensor networks'. Int. Conf. on Communications and Mobile Computing, Shenzhen, China, April 2010, pp. 142–146

[13] Jangra, A., Priyanka, S.: 'Securing LEACH protocol from sybil attack using jakes channel scheme (JCS)'. Int. Conf. on Advances in ICT for Emerging Regions, Sri Lanka Foundation Institute, Colombo, Sri Lanka, 2011, pp. 79–87

[14] Jan, M.A., Nanda, P., He, X., *et al.*: 'A sybil attack detection scheme for a centralized clustering-based hierarchical network'. Trustcom/BigDataSE/ISPA1, Helsinki, Finland, August 2015, pp. 318–325

[15] Jamshidi, M., Zangeneh, E., Esnaashari, M., *et al.*: 'A novel model of sybil attack in cluster-based wireless sensor networks and propose a distributed algorithm to defend it', *Wirel. Pers. Commun.*, 2018, **105**, (1), pp. 145–173

[16] Piro, C., Shields, C., Levine, B.N.: 'Detecting the sybil attack in mobile *Ad hoc* networks'. Securecomm and Workshops, Baltimore, MD, USA, August 2006, pp. 1–11

[17] Jamshidi, M., Zangeneh, E., Esnaashari, M., *et al.*: 'A lightweight algorithm for detecting mobile sybil nodes in mobile wireless sensor networks', *Comput. Electr. Eng.*, 2017, **64**, pp. 220–232
[18] Jamshidi, M., Ranjbari, M., Esnaashari, M., *et al.*: 'Sybil node detection in mobile wireless sensor networks using observer nodes', *Int. J. Inf. Vis.*, 2018, **2**, (3), pp. 159–165
[19] Jamshidi, M., Darwesh, A.M., Lorenc, A., *et al.*: 'A precise algorithm for detecting malicious sybil nodes in mobile wireless sensor networks', *IEEE Trans. Smart Process. Comput.*, 2018, **7**, (6), pp. 457–466
[20] Juels, A., Brainard, J.: 'Client puzzles: a cryptographic countermeasure against connection depletion attacks'. NDSS, 99, 1999, pp. 151–165
[21] Aura, T., Nikander, P., Leiwo, J.: 'DOS-resistant authentication with client puzzles'. Int. Workshop on Security Protocols, Berlin, Heidelberg, April 2000, pp. 170–177
[22] Dwork, C., Naor, M.: 'Pricing via processing or combating junk mail'. Annual Int. Cryptology Conf., Berlin, Heidelberg, May 1993, pp. 139–147
[23] Singh, V.P., Jain, S., Singhai, J.: 'Hello flood attack and its countermeasures in wireless sensor networks', *Int. J. Comput. Sci. Issues*, 2010, **7**, (11), pp. 23–27
[24] Bocan, V.: 'Threshold puzzles: the evolution of DOS-resistant authentication', *Periodica Politechnica, Trans. Autom. Control Comput. Sci.*, 2004, **49**, p. 63
[25] Dong, Q., Gao, L., Li, X.: 'A new client-puzzle based DoS-resistant scheme of IEEE 802.11i wireless authentication protocol'. 3rd Int. Conf. on Biomedical Engineering and Informatics (BMEI), Yantai, China, October 2010, pp. 2712–2716
[26] Ranjbari, M., Torkestani, J.A.: 'A learning automata-based algorithm for energy and SLA efficient consolidation of virtual machines in cloud data centers', *J. Parallel Distrib. Comput.*, 2018, **113**, pp. 55–62
[27] Esnaashari, M., Meybodi, M.R.: 'A cellular learning automata-based deployment strategy for mobile wireless sensor networks', *J. Parallel Distrib. Comput.*, 2011, **71**, (7), pp. 988–1001
[28] Jangirala, S., Dheerendra, M., Sourav, M.: 'Secure lightweight user authentication and key agreement scheme for wireless sensor networks tailored for the internet of things environment'. Int. Conf. on Information Systems Security, Cham, November 2016, pp. 45–65
[29] Dutta, P.K., Hui, J.W., Chu, D.C., *et al.*: 'Securing the deluge network programming system'. Proc. of the 5th Int. Conf. on Information Processing in Sensor Networks, ACM, Nashville, Tennessee, USA, April 2006, pp. 326–333
[30] Sobeih, A., Hou, J.C., Kung, L.C., *et al.*: 'J-Sim: a simulation and emulation environment for wireless sensor networks', *IEEE Wirel. Commun.*, 2006, **13**, (4), pp. 104–119