Detection of Node Capture under Asynchronous Sleep Mode Based on Recurrent Neural Network

Jintao Gu [1*], Jianyu Wang [1], Hao Sun [2]

[1]School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing, China

[2]School of Computer Science, Beijing University of Posts and telecommunications, Beijing, China

**Corresponding Author.*

Abstract

In recent years, wireless sensor networks (WSNs) have found numerous applications in industrial manufacturing and people's daily lives. However, security risks associated with the use of WSNs have also become increasingly pronounced. An attacker launching an internal attack on a WSN must first physically capture several nodes, i.e., take control of the target nodes by acquiring, cracking, and analyzing important information carried by the target nodes, thus laying the groundwork for subsequent attack steps. Therefore, physical node capture is a critical step in an internal attack on a WSN. Detecting behaviors that indicate physical capture of nodes provides an early warning of a network attack, allowing steps to be taken to prevent further attacks from being launched from the captured nodes. This paper proposes an RNN (recurrent neural network)-based detection method that can be used to detect node capture in WSNs with asynchronous sleep mode at an early stage (i.e., before captured nodes rejoin the network). Thus, the method enables early detection of network attacks. During the decision-making process, a common monitoring mechanism that relies on cooperation between neighboring nodes is employed to improve detection accuracy. The proposed method obtains the sensor nodes' states and makes a judgment with the help of RNN, achieving accurate detection of node capture under the condition of unsynchronized clocks. Simulation results demonstrate the proposed method's capability to achieve high detection accuracy.

Keywords*: wireless sensor network, network security, RNN*

I. Introduction

Thanks to their self-organizing capability, convenience, and low cost, wireless sensor networks are widely used in the sensing layer of the Internet of Things (IoT) suitable for a broad range of applications such as smart home, health monitoring, environmental monitoring, and battlefield data collection[1-3]. To ensure high accuracy in data collection, wireless sensor networks are usually deployed with several redundant nodes, and the sleep/wake mechanism is used to reduce the nodes' energy consumption[4]. In practical applications, sensor nodes are usually deployed in a harsh open environment, often lacking necessary physical protection. Moreover, the networks are generally not equipped with physical tamper-proof devices due to cost constraints[5]. Therefore, the open nature of their environment and wireless communication expose the nodes to various security threats. By obtaining all important confidential data stored in the nodes (including sensing data, codes, and keys) and modifying the program codes[6], the attackers can manipulate the sensor nodes to launch an internal attack on the network, thereby taking control of the network or damaging the network.

An internal attack on wireless sensor networks generally unfolds in three stages[7, 8]:

stage 1: Physically capture several nodes, and obtain and crack all the key confidential data stored in the nodes;

stage 2: Register the captured nodes or their clone nodes (also known as replicated nodes) in the original network and join the network communication;

stage 3: Manipulate the captured nodes to launch internal attacks such as selective forwarding attack, black hole attack, witch attack, energy consumption attack, and malicious data injection attack.

Researchers have developed several detection techniques suitable for different attack stages. Most techniques proposed in the literature are intended for detection in the second and third stages. Such techniques include, for example, node redeployment location detection[9], node clone detection[5, 10], selective forwarding attack detection[11], black hole attack detection[12], witch attack detection[13], Denial of Service attack detection, and malicious data injection detection[14]. Node capture should be detected as soon as possible to reduce the damage caused by the captured nodes to the network. Ideally, the captured nodes should be discovered as soon as the node capture occurs and be isolated from the network communication[15]. Such a node discovery action corresponds to the early detection of the physical node capture attack in the first stage.

This study develops a new method for detecting node capture attacks in wireless sensor networks. The method adopts a mechanism for scheduling and broadcasting node announcement messages in a unified manner to ensure that neighboring nodes have at least one opportunity to receive announcement messages during the sleep period. During the monitoring process, a mechanism relying on the neighboring nodes' cooperative decision-making effectively reduces the false alarm rate. The problem of low detection accuracy caused by unsynchronized clocks in the wireless sensor network is solved using a recurrent neural network (RNN).

II. Related Work

Physical node capture attack typically exploits sensors' programming and testing interfaces – such as the JTAG interface (used for chip testing and online system programming) – to compromise the sensors, effectively gaining control over them. Becher et al. [16] show that the time and conditions needed to accomplish a node capture attack vary depending on the degree of the required node control. Nevertheless, in an attack, the target nodes are necessarily cut off from normal network communication for a certain period, ranging from several minutes to several hours. Taking advantage of this common feature of node capture attacks, the existing node capture detection methods employ broadcasting beacon messages, hello messages, and heartbeat messages exchanged between nodes to monitor each node's survivability and detect captured nodes at the earliest possible time.

The CAT method[17] pairs nodes into node couples. Each node in a couple monitors the survivability of the other node by sending and receiving beacon messages. If one node fails to receive the beacon message sent by its pair node or receives a false beacon message, it concludes that the other node is experiencing a node capture attack and sends off an alarm. However, if both nodes in a node couple are captured simultaneously, they cannot detect the attack and send the alarm. Further, this method does not consider the node sleep mechanism. A node is in a sleep state misses the survivability announcement messages broadcasted by its neighboring node and cannot broadcast its announcement messages, which leads to a high false alarm rate.

The First Stage Detection (FSD) and Sink Enhanced FSD (SEFSD) methods [18] dictate that each node periodically broadcasts a greeting message to its neighboring nodes. Once the monitoring node fails to receive a greeting message, it starts counting consecutive messages that have not been received. When the count exceeds a certain limit, the monitored node is deemed as captured. The difference between the two methods lies in their response to captured node detection. FSD broadcasts alarm information to the entire network immediately after detecting an attack. In contrast, SEFSD sends an alarm to the base station (BS), and then BS sends alarm information to the whole network periodically with a fixed interval. SEFSD generates less communication overhead and consumes less power than FSD. However, the two methods suffer the drawback of relying on a single type of information. The monitoring node judges whether the monitored node has been captured based on the receipt of the declaration message from the monitored node without seeking confirmation from other neighboring nodes. Such a procedure results in a higher false alarm rate.

The Adaptive Early Node Compromise Detection (AdaptENCD) method[15] also utilizes heartbeat messages to

monitor the node survivability. This method uses different broadcasting frequencies flexibly for different clusters in a clustered network, thus promoting energy saving. This method is suitable for clustered networks. Its carefully designed intra-cluster node communication and key distribution/update mechanisms work well under different network operating conditions. However, the flexible use of message broadcasting frequency makes it difficult to detect node capture occurring in the same cluster accurately.

The method proposed by Ding et al.[17] considers nodes' sleep mechanism in its design of a node capture detection mechanism based on survivability monitoring. However, this approach incurs extra resource consumption as it requires an additional synchronization mechanism to ensure all nodes enter the sleep state and are awakened simultaneously, thus limiting its application scope. The method proposed by Hsin and Liu[18] dictates that the monitoring node takes into account the opinions of other nodes when making a decision (so-called decision-making based on local consensus). Still, this method does not consider the node sleep mechanism.

III. Framework

The detection system proposed in this paper can be divided into two functional modules (see Figure 1): the sensors state information extraction & collection module and the RNN-based calculation & decision module.

In the sensor state information extraction & collection module, each node monitors the message sending and receiving activities of all its neighboring nodes, based on which it forms a state transition sequence record for each neighboring node. In the RNN-based calculation & decision module, each node is assigned a detection node. The detection node inputs the message sequence into the RNN for training. The trained detection model is then embedded into the sensor node for detecting whether the subsequent message sequence is normal.

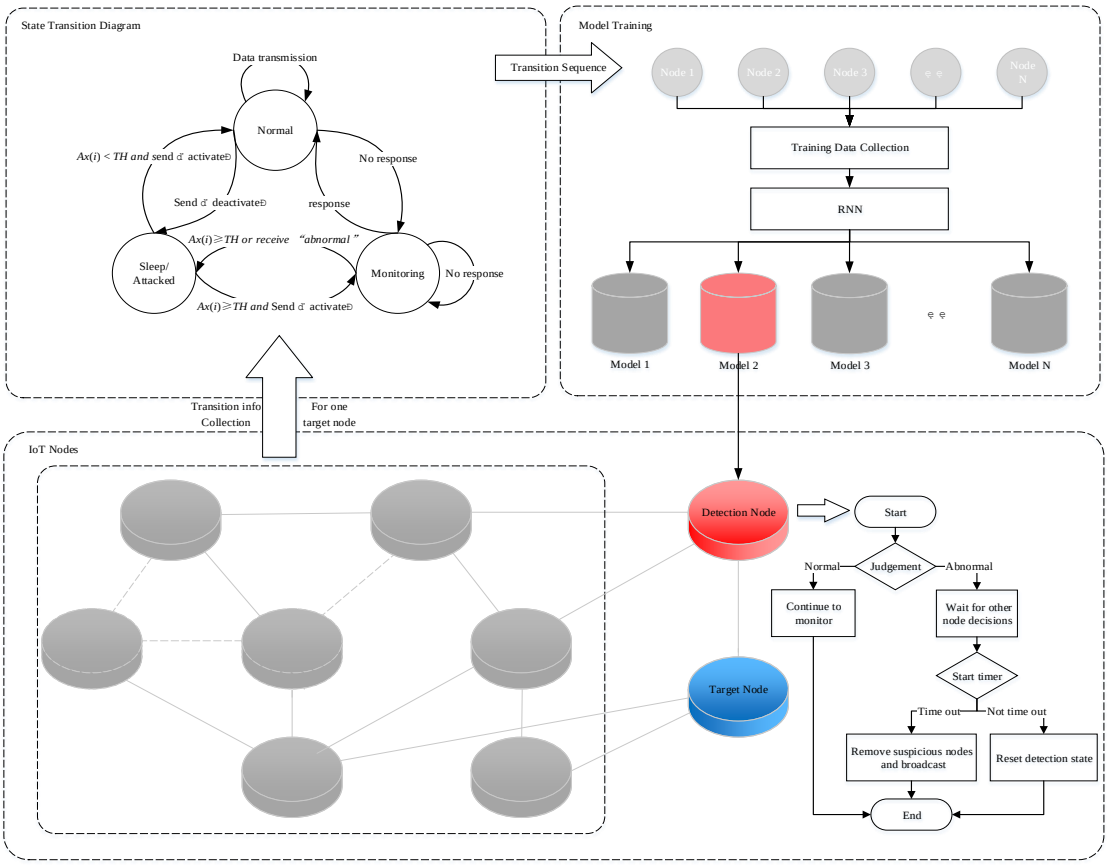

Figure 1 The node capture detection framework

The specific idea is as follows. First, the information regarding a certain target node in the IoT link and the nodes

that exchange data with this node (either adjacent or non-adjacent nodes) is extracted in a non-discriminative manner. The target node's state transition is judged based on the extracted information, and a state transition sequence is generated. Then the state transition sequence is input into the RNN model for training. For different target nodes, the corresponding detection models are created. The trained detection model is embedded in the detection node connected with the target node. The detection procedure is as follows:

1) When the target node does not respond to the messages sent by other nodes, the monitoring node makes a judgment on the target node's collected state sequence. If the sequence is judged to be a typical message sequence, the monitoring node continues to monitor and terminates the current judging state. If the sequence is judged to be an abnormal message sequence, the monitoring node starts a timer and waits for other nodes to refute the judgment.

2) If the timer expires and the monitoring node fails to receive verification messages from other detection nodes, the monitoring node removes the suspicious target node and broadcasts the information to the whole network. If the timer does not expire and the verification information is received, the monitoring node resets its detection state relating to the target node. Once the outlined actions are executed, the monitoring node terminates the judging state.

IV. The Method Details

4.1 Overview of the proposed method

In the node capturing process, the attacker must cut off the target nodes from the network for a specific time. To discover the attackers' node capture behavior in time, the proposed method puts a mechanism in place which enables monitoring the nodes' survivability.

4.2 Definition of node communication messages

Data transmission message: When node *i* continuously conducts normal data transmission, it is judged to be in a normal active state in which no interference information from other nodes is needed.

Wake-up message: Once node *i* finishes a sleep session and the timer T1 expires after wake-up, the node broadcasts a wake-up message to the neighboring nodes. The message format is <awake, Id_i, N_i>, which is denoted as $A(i)$. Id_i is the Id of node *i*, and N_i denotes the node *i*'s neighbor list.

Sleep message: Node *i* entering a sleep session broadcasts a sleep message to the neighboring nodes. The message format is <sleep, Id_i, N_i>, which is denoted as $A(i)$.

Inquiry message: When node *i* does not respond to messages and the monitoring node judges that the node is suspicious, it needs to broadcast an inquiry message to the neighboring nodes asking them to assist in the examination. The format of the broadcast message is <detect, Id_i, Id_j, N_i> and is denoted as $D(i)$. Here Id_i is the Id of node *i*, Id_j is the Id of the examined node *j*, and N_i is the neighbor list of node *i*.

Monitoring cancellation message: When node *i* receives the inquiry message about node *j* sent by other nodes, it checks its monitoring results in the current work cycle. If node *j* is in the normal state, node *i* replies to the inquiring node with the monitoring cancellation message. The format of the message is < safe, Id_i, Id_j >.

Isolation state message: When node *i* is judged to be captured, it is necessary to issue a warning to the whole network and remove the node. The format of the alarm message is < danger, Id_i, Id_j>.

ISSN: 0010-8189

4.3 Detailed procedure of the node capture detection

The procedure for early node capture detection is run in a distributed way among all nodes in the network. Each node makes a judgment during a sleep session based on its monitoring data and the neighboring nodes' feedbacks. The judgment is broadcasted to the whole network. When the judgment is uncertain, the node broadcasts an inquiry message to the surrounding nodes and decides whether the monitored node is abnormal based on the nodes' feedback. When the currently monitored node is judged to be in an abnormal state, the judgment will be sent to the base station through multi-hop routing. The base station notifies the whole network about the judgment periodically and takes necessary measures (e.g., isolate the captured nodes and update keys).

Each node is responsible for monitoring another node in its neighborhood while simultaneously being monitored by another neighboring node. Every node maintains a state information tag $S(s)$ for the node it monitors. $S(s)$ can be expressed as < S, id >, and the state set is S={normal, monitoring started, continuous monitoring, suspected}. The BS node is responsible for setting the abnormal alarm threshold TH for the whole network.

The procedures monitoring node (m) follows to judge the state of the monitored node (s) and the cooperative decision-making process are shown in Figure 2.

Normal state: No matter what state the monitored node s is in, as long as the monitoring node m receives any message from s during the working period or the other nodes judge the node as normal, the monitored node's state is set to normal. A message indicating that node s is in the normal state is broadcasted to the neighboring nodes at the end of the working period (i.e., immediately before entering the sleep state). Under normal conditions, there are three possible scenarios:

1) The monitoring node receives messages from other nodes indicating that node s is in the normal state. The only required action is to announce that node s is in the normal state before entering the sleep state.

2) The monitoring node receives a message from another node indicating that node s is in a suspicious state. It sends a suspicion cancellation message to the message source node.

Continuous monitoring state: When monitoring node m wakes up, there is no need to modify the monitored node's state if the neighboring node of the monitored node is in the continuous monitoring state. Normally, there are three possible scenarios:

1) Node m first checks the number of continuous monitoring cycles accomplished by node s. If the number of monitoring cycles is greater than or equal to TH, node m sets the state of node s to 'suspicious' and sends a detection message to the neighboring nodes.

2) If m receives any message from node s (including $A(s)$ message) during the working cycle, the node sends an inquiry message to other nodes and changes the state of s to 'normal'.

3) If m receives messages from other neighboring nodes during the working cycle indicating that node s is in the normal state, m changes the state of node s to 'normal'.

4) Suppose m fails to receive any message from node s during the working cycle while also not receiving any message from other nodes indicating node s is in the normal state. In that case, node m increases the count of continuous monitoring cycles for node s by one before entering the sleep state.

Sleep/abnormal state (SoA): When the monitoring node's timer expires, node s is forced to undergo an abnormal turn-off and enter the abnormal sleep state. In contrast, when node s receives the sleep instruction, it undergoes normal turn-off and enters the normal sleep state. When node s wakes up again, two possible scenarios may occur:

1) If the node was in the normal sleep state before turn-off, it enters the normal state.
2) If the node was in the abnormal sleep state before turn-off, it enters the continuous monitoring state.

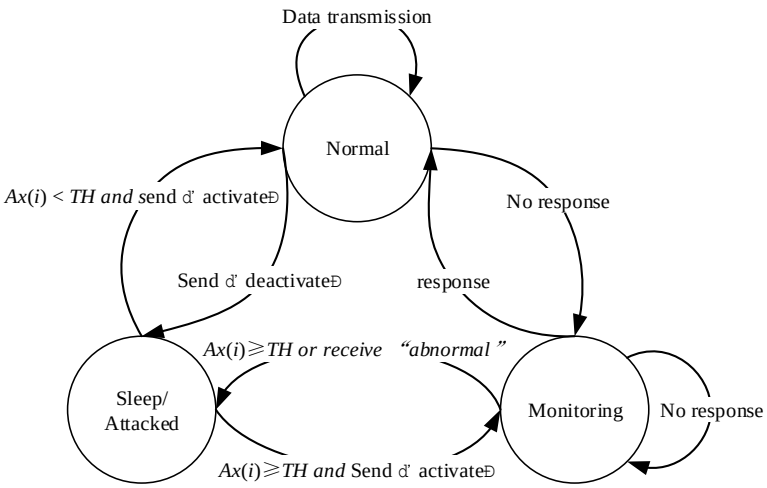

Figure 2 State transitions of monitored node i and monitoring node s

By enabling node *m* to perform active monitoring on neighboring nodes and using the unified mechanism of scheduling and broadcasting node announcement messages, the proposed method effectively detects node capture attacks under asynchronous sleep mode but also works well under synchronous sleep mode. It should be noted that each node *m* is both a monitoring node and a monitored node. In other words, each node is monitored by its neighboring nodes and follows the unified mechanism of scheduling and broadcasting node announcement messages.

When BS broadcasts the state judgment message about a certain node during the monitoring process, it indicates that the node is experiencing clock synchronization issues or is in an abnormal state. In both cases, the node should be isolated. For the nodes at the network edge or other special nodes located at relatively isolated places, BS should perform the task of judging whether the node has been captured. At the initial stage of network deployment, each node sends its neighbor list to BS. Thus, when BS receives the node capture alarm message, it notifies the whole network of the node capture periodically and updates each node's neighbor list by deleting the captured node from the lists. If a node has only one neighboring node and that node has been captured, BS also deems this node as a captured node because it has become an isolated node (which means no node is monitoring it).

4.4 Recurrent neural network

RNN is a neural network with a special structure that was put forward following the idea that human cognition is based on past experience and memory. RNN got its name due to the current sequence output being related to the previous outputs. Specifically, the network memorizes the previous information and applies it to the current output calculation. That is to say, the nodes between hidden layers are connected instead of disconnected, and the input of hidden layers includes not only the output of the input layer but also the hidden layer's output at the previous moment. The RNN's structure is illustrated in Figure 3.

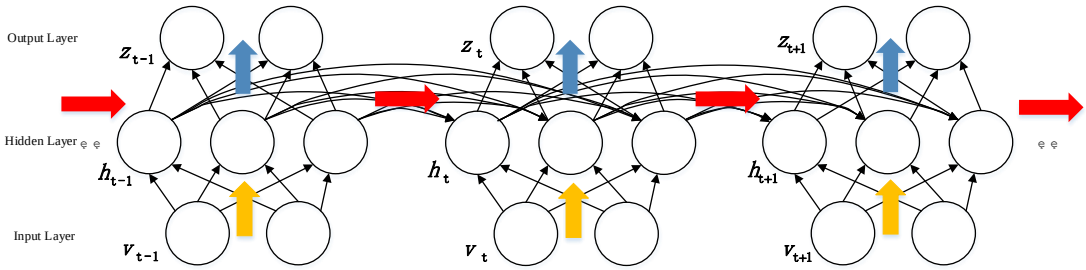

Figure 3 Structure of the recurrent neural network (RNN)

As noted, RNN is composed of an input layer, hidden layer, and output layer. The input layer is responsible for inputting node energy information into the neural network. The main task of the hidden layer is calculating input data. Finally, the output layer outputs the prediction results.

V. Experiment

The conducted simulation experiment aimed to evaluate the effectiveness and communication overhead of the proposed node capture detection method. The detection rate, false-positive rate, false-negative rate, and communication overhead during the detection process were extracted. Further, the performance of the proposed method before and after embedding the RNN algorithm was studied. The experiment used OMNET++ for simulation. Two typical node capture detection algorithms, CAT[7] and SEFSD[8], were also simulated under the same conditions for comparison. Part of the experiment related to deep learning was conducted in the Ubantu14.04LTS environment. Python 3.5.4 and Keras 2.1.2 were used to construct the neural network framework, and Google's open-source tensorflow1.4.1 was used as the back-end computing framework. The CPU of the server was Inter (r) Xeon (r) CPU E5-2637v4 @ 3.50ghz, and the GPU was TITAN(X)(Pascal).

5.1 Description of the experiment and verification indicators

In the experiment, 200 nodes and one base station were randomly and evenly deployed in a 200 m×200 m site. Each node's communication range was 40 m. The node capture was simulated by deactivating the nodes' communication function. The highest proportion of captured nodes was set to 25%, and the captured nodes were selected randomly. After the node deployment was completed, a random short delay was imposed on each node before entering a work cycle to simulate the asynchronous sleep state. Considering that attackers with high skills and the right equipment need only a short time to compromise nodes through programming a test interface, the time required for node capture (p_c) was set to 300 s. The sleep scheduling period was set to 60 s (the working period p_w was 40 s, and the sleep period p_s was 20 s). The step timer T_2 set by the node for each neighbor node was 10 s. To ensure the normal RNN function, the data acquisition, annotation, and training processes were carried out first for one hour. The proportion of captured nodes was 40% to ensure the balance between positive and negative samples in the training data. During initiation, the clock cycles of 20% nodes were set with certain errors, and the clock cycles of 10% nodes were set to different values. The compiled training data set is described in Table 1.

Table 1 The number and proportion of samples in the training data

	Number of samples	Proportion of samples
Total samples	714,682	100%
Normal samples	287,542	40.25%
Captured samples	287,697	40.25%
Samples not on time	139,443	19.50%

The verification was carried out using the following metrics:

$$P = \frac{TP}{TP + FP} \times 100\%,$$

$$R = \frac{TP}{TP + FN} \times 100\%,$$

where P represents precision, indicating the fraction of actual attacks among the detected ones. Recall (denoted R) is the proportion of node captures that have been detected. TP stands for true positive, FP is false positive, and FN denotes false negative.

5.2 RNN performance and data analysis

To ensure high performance of the proposed detection method, it is first necessary to verify the RNN's performance in the algorithm. The training adopts 40% cross-validation, and the model's obtained precision and recall rate are shown in Tables 2–4.

Table 2 Training results on normal nodes

	Dataset1	Dataset2	Dataset3	Dataset4
Validation number	72,000	72,000	72,000	72,000
Error number	236	184	339	278
P	99.52%	99.63%	99.69%	99.73%
R	99.67%	99.74%	99.53%	99.61%

Table 3 Training results on nodes with unsynchronized clock

	Dataset1	Dataset2	Dataset3	Dataset4
Validation number	72,000	72,000	72,000	72,000
Error number	452	531	629	553
P	99.40%	99.41%	99.34%	99.39%
R	99.37%	99.26%	99.13%	99.23%

Table 4 Training results on captured nodes

	Dataset1	Dataset2	Dataset3	Dataset4
Validation number	34,861	34,861	34,861	34,861
Error number	216	347	290	275
P	99.45%	99.21%	99.54%	99.37%
R	99.38%	99.00%	99.17%	99.21%

The tables show the precision and recall rate results obtained when RNN was employed to detect node capture in three scenarios. It can be seen that RNN achieves high accuracy and recall rate in two scenarios: when there are captured nodes and when the clocks are out of sync. These results can be attributed to the sensor nodes' working mode. As the sensors can only work in a very limited variety of modes, high accuracy can be achieved even though the input data is huge. The proposed detection method dictates that a sensor node is isolated immediately once the

base station deems it as having an unsynchronized clock out or being captured. Therefore, if RNN cannot achieve high accuracy, many sensor nodes are mistakenly considered problematic and removed from the sensor network, causing the problem of an insufficient number of sensor nodes. It can be seen from the experimental data that the trained RNN model provides powerful support to the proposed detection method, effectively performing the node detection when the clocks are unsynchronized and detecting the node captures.

5.3 Simulation results and analysis

An experiment was carried out to compare the detection effects before and after embedding the RNN algorithm into the proposed method. The specific experimental results are shown in Figures 4 and 5.

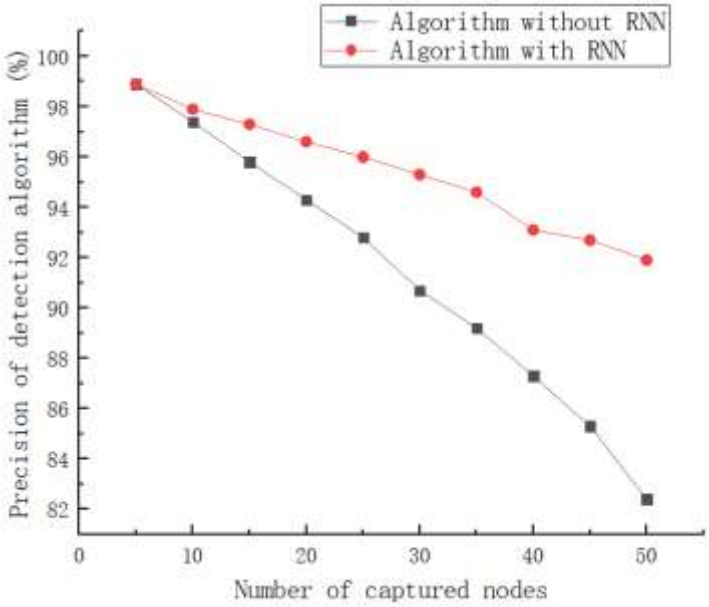

Figure 4 Prediction results before and after embedding the RNN algorithm

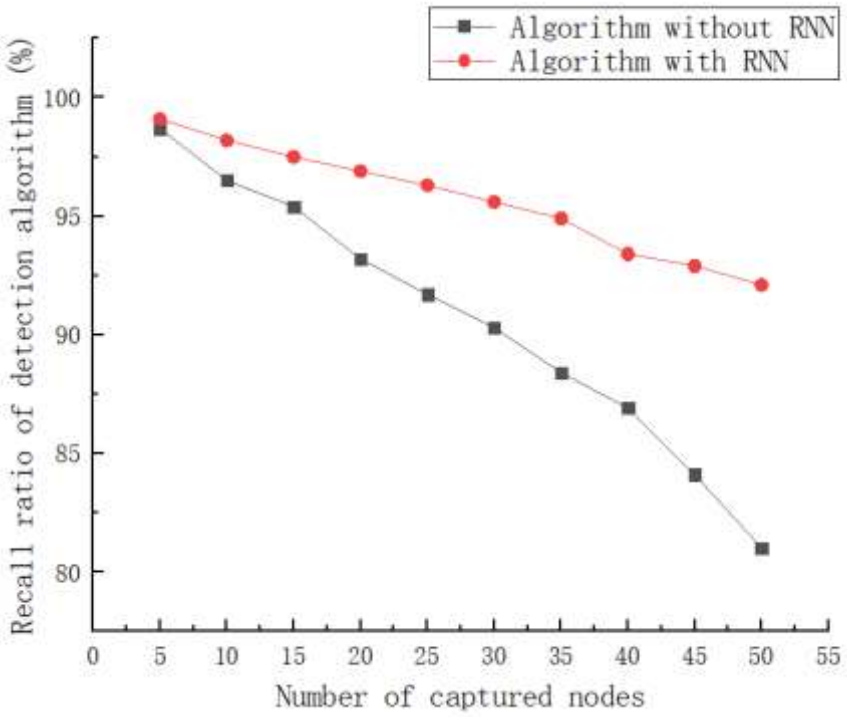

Figure 5 Recall rate results before and after embedding the RNN algorithm

It can be seen from the figures that embedding RNN into the proposed method improves the detection precision and recall rate effectively. On the one hand, embedding RNN enables effective node detection when the clocks are out of sync. On the other hand, the RNN's high precision and recall rate can improve the detection method's performance and effectively supplement the proposed method when the number of captured nodes is large.

VI. Conclusion

The existing methods for node capture detection in wireless sensor networks do not consider the nodes' asynchronous sleep. Thus, this paper proposes a method for detecting node capture attacks under the asynchronous sleep mode, consequently enabling early detection of the network attacks. The proposed method ensures that the monitoring node receives the declaration messages from the neighboring nodes through a unified mechanism of scheduling and broadcasting node declaration messages and improves the detection performance by combining the local collaborative decision-making mechanism with the RNN algorithm. Simulation results demonstrate the proposed method's effectiveness in counteracting asynchronous sleep on node capture detection, improving the detection rate of node capture attacks, and significantly reducing both the false-negative and false-positive rates.

References

[1] Y. Zhang, S. He, J. Chen, "Data gathering optimization by dynamic sensing and routing in rechargeable sensor networks," Sensor, Mesh & Ad Hoc Communications & Networks, vol. 24, no. 3, pp. 273-281, 2013.

[2] A. Liu, J. Ren, X. Li, et al., "Design principles and improvement of cost function based energy aware routing algorithms for wireless sensor networks," Computer Networks, vol. 56, no. 7, pp. 1951-1967, 2012.

[3] L. Jiang, A. Liu, Y. Hu, et al., "Lifetime maximization through dynamic ring-based routing scheme for correlated data collecting in WSNs," Comput Electr Eng, vol. 41, no. 1, pp. 191-215, 2015.

[4] H.S. Abdelsalam, S. Olariu, “Toward adaptive sleep schedules for balancing energy consumption in wireless sensor networks,” IEEE Transactions on Computers, vol. 61, no. 10, pp. 1443-1458, 2012.

[5] N. Shashidhar, C. Kari, R. Verma, “The efficacy of epidemic algorithms on detecting node replicas in wireless sensor networks,” J Sens Actuator Netw, vol. 4, pp. 378-409, 2015.

[6] C. Lin, G. Wu, C.W. Yu, et al., “Maximizing destructiveness of node capture attack in wireless sensor networks,” Journal of Supercomputing, vol. 71, no. 8, pp. 3181-3212, 2015.

[7] X. Lin, “CAT: Building couples to early detect node compromise attack in wireless sensor networks” Proc of Global Telecommunications Conference, Honolulu: IEEE Press, pp. 1-6, 2009.

[8] W. Ding, Y. Yu, S. Yenduri, “Distributed first stage detection for node capture,” Proc of Global Telecommunications Conference, Miami, FL: IEEE Press, pp. 1566-1570, 2010.

[9] H. Song, L. Xie, S. Zhu, et al., “Sensor node compromise detection: the location perspective,” Proc of International Conference on Wireless Communications and Mobile Computing, New York: ACM Press, pp. 242-247, 2007.

[10] Y. Zeng, J. Cao, S. Zhang, et al., “Random-walk based approach to detect clone attacks in wireless sensor networks,” IEEE J Select Areas Commun, vol. 28, no. 5, pp. 677-691, 2010.

[11] N.E. Rikli, A. Alnasser, “Lightweight trust model for the detection of concealed malicious nodes in sparse wireless ad hoc networks,” International Journal of Distributed Sensor Networks, vol. 12, no. 7, pp. 1-16, 2016.

[12] Y. Liu, M. Dong, K. Ota, et al., “ActiveTrust: secure and trustable routing in wireless sensor networks,” IEEE Trans on Information Forensics and Security, vol. 11, no. 9, pp. 2013-2027, 2016.

[13] S. Abbas, M. Merabti, J.D. Llewellyn, et al., “Lightweight sybil attack detection in MANETs,” IEEE Systems Journal, vol. 7, no. 2, pp. 236-248, 2013.

[14] S.M. Nam, T.H. Cho, “Context-aware architecture for probabilistic voting-based filtering scheme in sensor networks,” IEEE Trans on Mobile Computing, vol. 16, no. 10, pp. 2751-2763, 2017.

[15] A. Al-Riyami, N. Zhang, J.Keane, “An adaptive early node compromise detection scheme for hierarchical WSNs,” IEEE Access, vol. 4, pp. 4183-4206, 2016.

[16] A. Becher, Z. Benenson, M. Dornseif, “Tampering with motes: real-world physical attacks on wireless sensor networks,” Proc of the Third International Conference on Security in Pervasive Computing, York, UK: ACM Press, pp. 104-118, 2006.

[17] W. Ding, Y. Yu, S. Yenduri, “Energy saving by centralized sleep in early detection of captured nodes,” IEEE International Workshop on Robotic and Sensors Environments, Phoenix, AZ: IEEE Press, pp. 1-6, 2010.

[18] C. Hsin, M. Liu, “Self-monitoring of wireless sensor networks,” Computer Communications, vol. 29, no. 4, pp. 462-476, 2006.

[19] Butun I, Morgera S D, Sankar R. A Survey of Intrusion Detection Systems in Wireless Sensor Networks. IEEE Communications Surveys & Tutorials, 2014, 16(1): 266-282.