IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received September 28, 2020, accepted September 30, 2020, date of publication October 6, 2020, date of current version October 19, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3029191

RPL Attack Detection and Prevention in the Internet of Things Networks Using a GRU Based Deep Learning

SEMIH CAKIR[1], **SINAN TOKLU**[2], **AND NESIBE YALCIN**[3]

[1]Department of Electrical, Electronics, and Computer Engineering, Institute of Science, Düzce University, 81010 Düzce, Turkey
[2]Department of Computer Engineering, Düzce University, 81010 Düzce, Turkey
[3]Department of Computer Engineering, Bartın University, 74110 Bartin, Turkey

Corresponding author: Semih Cakir (semih.cakir@beun.edu.tr)

ABSTRACT Cyberattacks targeting Internet of Things (IoT), have increased significantly, over the past decade, with the spread of internet-connected smart devices and applications. Routing Protocol for Low-Power and Lossy Network (RPL) enables messages to be routed between nodes for the Wireless Sensor Network in the network layer. RPL protocol, which is sensitive and difficult to protect, is exposed to various attacks. These attacks negatively affect data transmission and cause great destruction to the topology by consuming the resources. Hello Flooding (HF) attacks against RPL cause consumption of constrained resources (memory, processing and energy) in nodes. Therefore, in this study, a Gated Recurrent Unit network model based deep learning has been proposed to predict and prevent HF attacks on RPL protocol in IoT networks. The proposed model has been compared with Support Vector Machine and Logistic Regression methods, and different power states and total energy consumptions of the nodes have been taken into consideration and experimented with. The results confirm the promised and expected performance from the model in terms of source efficiency and IoT security. In addition, attack detection has been carried out with a much lower error rate than literature studies for HF attacks from RPL flood attacks.

INDEX TERMS Deep learning, gated recurrent unit, hello flooding, Internet of Things.

I. INTRODUCTION

Internet of Things (IoT) refers to devices, machines and software that communicate with each other when considered in a system, on the other hand everything connected to the internet [1]. The interaction of objects that communicate and transfer data, which are connected to the Internet via 6LoWPAN (IPv6 over Low-Power Wireless Personal Area Networks) is used to improve the quality of life (e.g. smart cities, smart buildings, smart cars) and increase job opportunities [2], [3]. However, due to its ad-hoc and limited resource structure, IoT systems are very vulnerable to attacks. Generally, attacks target the usability and energy consumption of a node connected to a heavy data stream. Attack detection systems are one of the security measures and are crucial in an IoT ecosystem.

The primary function of attacks on Routing Protocol for Low-Power and Lossy Network (RPL) based IoT networks,

The associate editor coordinating the review of this manuscript and approving it for publication was Firooz B. Saghezchi.

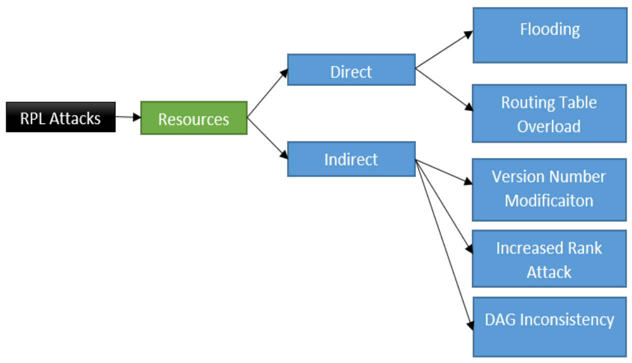

FIGURE 1. Well known RPL attacks against resources [4].

the most common type encountered in the literature, is to route messages between nodes. The most known RPL attacks targeting resource consumption [4] are given in Fig. 1. The purpose of source-side attacks is that the malicious node, which is harmful in our highly sensitive network structure, creates energy consumption, process crowd, and excessive memory density to disturb the stability of the Quality of Service (QoS) of the network. Flood attacks, one of the most known attack types, aim to render the nodes dysfunctional by damaging the network topology with the help of the malicious node. They perform this by transmitting or broadcasting DODAG Information Solicitation (DIS) messages.

Hello Flooding (HF) attacks are the RPL protocol attacks, which are not emphasized enough in terms of the energy consumption in nodes in the literature. The values of the Central Processing Unit (CPU; used for calculations data processing), Low Power Mode (LPM; idle device waiting for events), Radio Transmission (Tx; data transmission), Radio Reception (Rx; data reception) and Total Energy (TE; total energy consumption) power states that affect the energy of the network topology are calculated and the changes in the energy after the attacks are observed in this study.

The fact that significant amount of data is received from devices that lack resources and computing abilities has rendered the classical methods ineffective and led to the emergence of new systems [5], [6]. Therefore, it is reported in the literature that machine learning methods are more useful in terms of interpreting data in IoT attacks and making accurate predictions. For this reason, unlike similar studies for real-time attack detection on devices in the IoT ecosystem, the nodes have been classified using GRU (Gated Recurrent Unit)-based deep learning method with Recurrent Neural Network (RNN) architecture and achieved a high accuracy of 99.96%. On the other hand, 3 different datasets, whose designations start with "SSN", have been created as a result of the simulations conducted in Contiki OS / Cooja Simulator. In summary, the main contributions of this study are as follows:

a) "A GRU-based deep learning approach" is used to classify malicious nodes in HF attack detection and also to protect the constrained energy sources.

b) Each dataset, named "SSNx", includes CPU, LPM, Tx, Rx and TE data calculated for each node, and can be used to prevent attacks after classification. GRU is proven to yield more dominant results than Support Vector Machine (SVM) and Logistic Regression (LR). The packet of the node with a value received outside the specified value range will be dropped from the network after checking the threshold value of the properties calculated within the attack prevention method.

c) "Detection of HF attacks with a high accuracy rate" that can inspire attack detection and prevention methods in other IoT ecosystems.

The remainder of this paper is organized as follows: next section presents current studies made on IoT attacks, their detection and prevention approaches. Section III gives detailed information about HF attacks on RPL Protocol, deep learning-based attack detection, simulation of IoT ecosystem and dataset used in the experiment. The proposed model is explained in Section IV and the results are summarized and discussed in Section V. Conclusions are drawn in the final section.

II. RELATED STUDIES AND BACKGROUND

Integrity, reliability, confidentiality, and security are significant parameters for the sustainability of a system. IoT has been the focus of attention with its recent popularity, but has brought some problems along with it. Security is at the forefront of these problems [6]–[8]. In the classification of attacks, HF attacks are a type of routing attacks and are carried out in the network layer. Sinkhole, Sybil, Rank, HF, Spoofing, Blackhole, and Wormholes are known as common attacks in routing services [9].

Considering resource consumption, flood attacks come to the forefront. Although HF attacks are among the attacks that are effective, the energy consumption calculation of resources when detecting the malicious nodes has been the subject of not many studies in the literature. Therefore, this study will contribute novelty to the literature with the methods proposed and the effective attack prevention system recommended.

A. RPL AND RESOURCE ATTACKS

1) 6 LoWPAN (IPv6 OVER LOW-POWER WIRELESS PERSONAL AREA NETWORKS

There are insufficient IPv4 addresses and it creates conflicts in the current communication system due to increase in the number of IoT devices. IPv6 provides billions of unique IP addresses for the IoT ecosystem with 128-bit addressing [10]. 6LoWPAN[11] protocol is a communication protocol of IEEE 802.15.4 [12] standards that provides minimum resource consumption, long battery life, and high data capacity.

2) RPL (IPv6 ROUTING PROTOCOL FOR LOW-POWER AND LOSSY NETWORKS

RPL protocol [13] is a dynamic, distance - vector protocol and seeks to find paths between nodes on the network using routing protocols [14]. Its key features are automatic configuration, self-healing and loop avoidance, and its main function is to direct data traffic with minimum energy consumption and minimum loss of packets. In addition, it can be used in point-to-point (P2P), multipoint-to-point (MP2P), and point-to-multipoint (P2MP) topology types. In the tree-based topology, the data flow is root-to-nodes or vice versa. The position where each node is located in the topology is called "Rank". Nodes are positioned in the topology based on their levels calculated with the help of an "objective function". This function uses algorithms that calculate the quality of the route with the help of metrics.

Main concepts used in RPL protocol;

Destination Oriented Directed Acyclic Graphs (DODAG): The nodes are interconnected according to a specific topology that incorporates tree and mesh topologies called DODAG [4]. This is a special kind of Directed Acyclic Graphs (DAG) where each node wishes to reach a single destination.

DODAG Information Object (DIO): This message is used to discover new nodes, transmit configuration and communicate. If a node receives a DIO message, it also emits a DIO message, taking into account the rank value and link metrics of the incoming node. In this way, the parent-child relationship is also determined.

DODAG Information Solicitation (DIS): It can be considered as a message to discover neighbors. A node, not included in the DODAG network, broadcasts a DIS message. It sends it to ask "Are there any DODAGs?".

DODAG Advertisement Object (DAO): It is sent by a child to its parent. This message includes a request to allow the child to join a DODAG. DAO-ACK: This message includes "Yes" or "No" response from parent or root to the child.

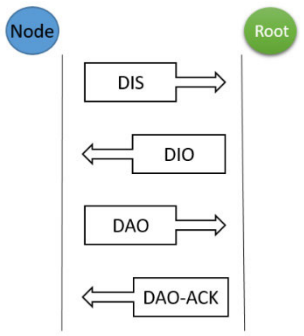

FIGURE 2. RPL control messages from node to root [15].

The message relationship between the root node and a new node that wants to join DODAG is given in Fig. 2 [15].

3) ROUTING AND RESOURCE ATTACKS

Routing protocol resource attacks in wireless sensor networks (WSN) are typically based on consuming the limited resources (energy, memory or process density) of the nodes in the topology through unnecessary operations. Thus, the life of a network decreases and loses its functionality. As shown in Fig. 3, resource attacks can be classified in 3 groups as wireless network attacks, routing attacks and denial of service attacks [4].

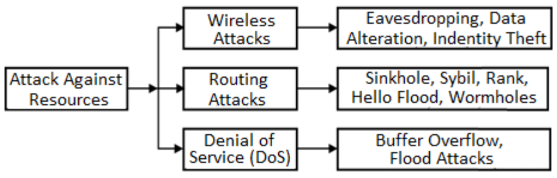
Attack Against Resources
Wireless Attacks
Eavesdropping, Data Alteration, Indentity Theft
Routing Attacks
Sinkhole, Sybil, Rank, Hello Flood, Wormholes
Denial of Service (DoS)
Buffer Overflow, Flood Attacks

FIGURE 3. Classification of attacks.

In HF attacks, each node creates the neighborhood list by sending a packet with a "Hello" message to their neighbors. Any malicious node in the network ecosystem attempts to strike network traffic by sending "Hello" messages to many othe nodes, and imposes itself to other nodes, transmits the messages that will go through other nodes all the way to the server and performs the transmission [16]–[19]. The malicious node set in DODAG increases the network traffic and sends messages to the nearby nodes more than usual. When the HF attack begins, it increases the resource consumption by directing the data traffic on the neighboring nodes via itself. In the network ecosystem where huge data traffic occurs, the energy consumption of the limited nodes increases and data losses occur. The repair mechanism that corrects the RPL's data flow resolves this problem after a certain period, but the nodes around the malicious node will continue to be affected by this attack.

B. DETECTION AND PREVENTION APPROACHES

Attacks on WSN are carried out in IoT and their protocol structures, designs, and messages are different. Therefore, prevention or reduction of the attacks may differ. In other words, the methods applied to one type of attack may be invalid for others [20].

There are many anomaly-based, signature-based and encryption methods, machine learning, deep learning, neural network types and classification methods are used for detection and prevention of IoT attacks in the literature. Current studies using these methods and outputs are presented in Table 1. Cryptographic and non-cryptographic methods are used for HF detection but they are not suitable for the resource-constrained IoT ecosystem [21]. New detection methods are needed in terms of limited storage space, processing capacity, and energy source. Attack prevention systems are generally in the form of mitigating or minimizing the attacks.

III. DEEP LEARNING BASED RPL ATTACK DETECTION

"Deep learning", first used by Igor Aizenberg and colleagues, is a subfield of machine learning and is based on Artificial Neural Networks (ANNs). The distinction of deep learning from ANN is the hidden layers in its structure. Consecutive layers take the output of the previous layer as input and its structure is based on learning the representation of the data [27]–[29]. Deep learning methods give better results than traditional data processing techniques for the data in very large sizes [1], [5], [6], [28], [29], [36]. Therefore, deep learning methods are suitable for big data created by IoT devices. Traditional authentication methods or RSS-based approaches cannot always provide security for resource-restricted IoT devices in HF attacks. Detection methods for HF attacks (bidirectional verification technique [30]), prevention solutions (identity verification protocol [31], multi-path multi-base station routing [30] and μ-Tesla [32]), and both attack detection and prevention approaches (deep learning methods [1], [28], [29]) have presented for HF attacks on IoT devices in the literature.

In the proposed methodology, the Cooja simulator in the Contiki operating system is used for simulation scenarios, and the data packets of the attacker and normal nodes have been created in Zonguldak Bülent Ecevit University Kdz. Eregli Vocational School. Total energies of nodes, CPU, LPM, Tx and Rx power values have been calculated from the IoT routing attacks based on the characteristics of the HF attacks

TABLE 1. Summary of flooding attacks and countermeasures on RPL.

Reference	Technologies	Objectives	Impact		Time Complexity
[1]	A deep-learning for detection of Decreased Rank, Hello Flood and Version Number routing attacks	To detect decreased rank, hello-flood and version number attacks with high accuracy and precision.	**Benefit:** can detect attacks with high accuracy and model has reached up to 99.5% accuracy	**Drawbacks:** lack of information about any machine learning method and inappropriate for constrained devices	Just given accuracy
[3]	A proposed novel IDS implemented in Contiki OS/ Cooja Simulator and RSSI	To detect a wormhole attack and any attacker using neighbor information	**Benefit:** can identify the attack completely	**Drawbacks:** inappropriate for constrained devices	Not given
[17]	RSSI (Received Signal Strength Indicator) and time threshold based AODV-HFDP – implemented in NS-2	To detect HF attacks using signal strength and distance between two nodes	**Benefit:** can detect attacks and improving the performance of the network	**Drawbacks:** inappropriate for constrained devices	Not given
[19]	A proposed RSSI and client puzzles method	To detect HF attacks, the nodes are grouped as a friend or a stranger using signal power measurements: if any node sends *a* times hello messages then it has to solve more difficult puzzles in the b^{th} level.	**Benefit:** can detect HF attacks	**Drawbacks:** inappropriate for constrained devices and when the attack initiated by several colluding nodes, HF attacks cannot be detected. In addition, the puzzles cause computational overhead and increase the complexity of a dataset.	Not given
[22]	IDS based (Anomaly detection techniques)- Neighborhood – implemented in TOSSIM Simulator- TinyOS.	To calculate the threshold of RSSI based on the difference between the signal strength of two neighbors. So, it is identified as malicious node or not in hello flood. If a node is malicious, it's removed from the neighbor list and added to the blacklist.	**Benefit:** can detect attacks	**Drawbacks:** inappropriate for constrained devices	Not given
[23]	RSSI values	To have a defense system against Hello Flood attacks with the help of an RSS based approach	**Benefit:** can detect attacks	**Drawbacks:** inappropriate for constrained devices	Not given
[24]	A network-based DoS attack detection, IDS architecture on network framework	To detect DoS attacks based on 6LoWPAN.	**Benefit:** can detect DoS attacks and may help detect more complicated attacks.	**Drawbacks:** inappropriate for constrained devices	Not given
[25]	Cooperative based FAIS - IDS	To defend against to flooding packets with immunology based FAIS and cooperative fuzzy Q-learning defense mechanism	**Benefit:** can detect flooding attacks	**Drawbacks:** inappropriate for constrained devices	Not given
[26]	D-FICCA – IDS mechanism was modified with the DB clustering algorithm for optimum clustering.	To enhance the accuracy of detection of malicious nodes	**Benefit:** can detect flooding attacks	**Drawbacks:** inappropriate for constrained devices	Not given
[45]	Machine Learning approach based on Kernel Density Estimation (KDE) technique for anomaly detection.	To detect the Version Number, HF and Black Hole attacks using KDE technique. Authors have counted DIS, DIO and DAO packets which are created on every node in topology.	**Benefit:** can detect HF attacks with average 90% accuracy.	**Drawbacks:** the only "counted DIS, DIO and DAO packets" feature is not convenient for detecting HF attacks.	O (n)
This study	A deep-learning based model for detection and prevention of RPL attacks in IoT networks.	To detect HF attacks with a high accuracy rate, to drop malicious nodes from the network and to protect the constrained energy sources	**Benefit:** can detect HF attacks with high accuracy of 99.96%.	**Drawbacks:** scalability problem when the number of nodes is increased	O (n)

and recorded in a dataset. The learning algorithm has been implemented with the support of KERAS [33] Tensorflow from Python libraries.

A. DEEP LEARNING ARCHITECTURE

RNN is a type of ANN where the units in the system are connected to each other with a loop and based on the logic of receiving raw data in a certain order [29]. RNN has a short-term memory problem [34]. If a time series data is quite long, it may be difficult to remember the previous data during the move of the information from the previous step to the next step, and it becomes difficult to forward the information. Therefore, the information that is important in the prediction can be missing. LSTM (Long Short – Term Memory), developed by Hochreiter and Schmidhuber in 1997 to solve this problem [35], is a special type of RNN that can learn in long arrays, However, its complex structure and time-consuming analysis, compared to neural networks, led to the emergence of the GRU in 2014. Features such as fast training, simplified structure, and being easy-to-analyze make GRU stand out compared to LSTM [36].

GRU has a similar structure when compared to LSTM units. Unlike LSTM, GRU has a reset and update gateway instead of the entrance, exit and forget gate. The purpose of the reset gate is how the new entry will be combined with the previous memory, and the update gate describes how much of the previous memory will be stored. These vectors decide what information is to be transferred as the output and can be trained to remove information that is irrelevant for prediction. Briefly, it can be learned which of the data in the series is important to store and delete and also the learned information can be used for predictive purposes. The general structure and architecture of GRU neural networks are illustrated in Fig. 4 and Fig. 5, respectively.

The terms shown in Fig. 5 are explained as follows [39], [40]:

Update gate helps determine how much of the past information for the model needs to be passed into the future. With this feature, it prevents the model from copying all the information from the past. The model may decide to copy all the information from the past and gradient can eliminate the risk of eliminating the problem. Equation (1) is used to calculate the update gate z_t for time step t.

$$z_t = \sigma \times (W_z \times h_{t-1} + x_t \times U_z) \tag{1}$$

Here, x_t is the input vector and h_{t-1} holds the output for the previous time step $t-1$. When x_t and h_{t-1} are attached into the model unit, they are multiplied by their weights U_z and Wz, respectively. The multiplication results are summed and then a sigmoid activation function σ is applied to match the results between 0 and 1.

Reset gate r_t is used to decide how much of the past information in the model is forgotten and determined by (2)

$$r_t = \sigma \times (W_r \times h_{t-1} + x_t \times U_r) \tag{2}$$

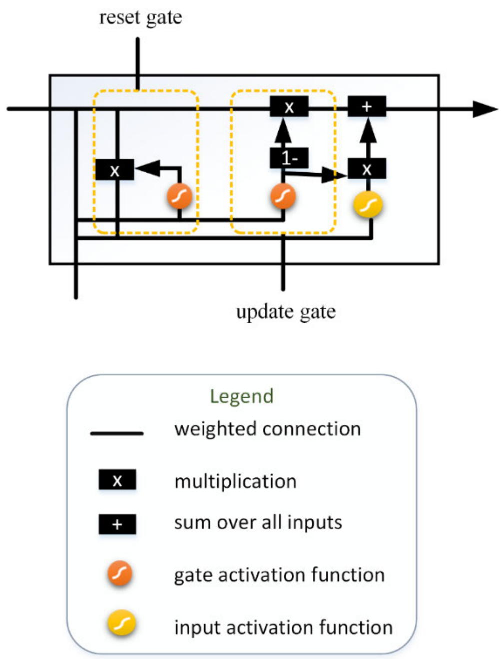

FIGURE 4. Structure of GRU cells [37].

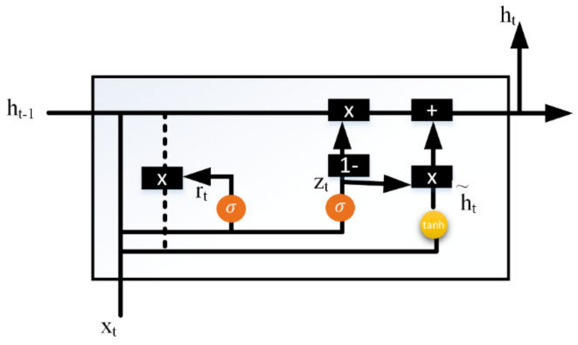

FIGURE 5. The GRU architecture [38].

where, h_{t-1} and x_t are multiplied by their corresponding weights W_r and U_r, respectively. Then σ sigmoid function is applied on the sum of the results.

Current memory content: A new memory content $\hat{h}_t$ uses the reset gate to store the relevant information from the past. The formula is given in (3)

$$\tilde{h}_t = tanh \times (W_h \times (h_{t-1} \times r) + U_c \times x_t) \tag{3}$$

where the input x_t is multiplied with the weight U_c, h_{t-1} is multiplied with the reset gate r and with the weight W_h. That will assign what to extract from the previous time steps. And then the results are summed up and nonlinear activation function *tanh* is finally applied.

Last memory at the current time: The model needs to calculate the h_t vector, which holds the information for the current unit and passes it to the model. An update gate is required for this. It determines what to gather from the current memory content $\hat{h}_t$ and what from the previous steps h_{t-1} and calculates it with the help of (4).

$$h_t = \left(z_t \times \tilde{h}_t\right) + ((1 - z_t) \times h_{t-1}) \tag{4}$$

B. SIMULATION OF IoT ECOSYSTEM AND SSN

The variety of use of WSN technology in different environments has enabled it to take place in the IoT ecosystem. Possible aspects of this technology can be examined through the simulators that offer real-life environments. Contiki - Cooja simulator [41] is an ideal tool to develop the simulation platforms for RPL and WSN. It is open for research purposes and has been chosen thanks to its functionality and scalability features. The simulation control window supports start, pause, reload and stop buttons for the simulation in addition to the simulation time and speed of the running simulation. Parameters of the Cooja simulator have been designated for different scenarios (1, 2 and 3) and are presented in Table 2.

TABLE 2. Simulation parameters.

Parameter	Value / Setting
Scenario	1 / 2 / 3
simulation tool	Instant Contiki 3.0 Cooja Simulator
mote type	Sky mote
source code motes	Contiki/examples/ipv6/rpl-udp/udp-server.c Contiki/examples/ipv6/rpl-udp/udp-client.c
source code malicious	Contiki/examples/ipv6/rpl-udp/hfmalicious.c
simulation run time	36000 s
total number of motes	6 / 11 / 16
number of sink motes	1 / 1 / 1
number of legitimate motes	4 / 9 / 12
number of HF malicious motes	1 / 1 / 3
radio medium transmission range	UDGM: Distance loss 50 m
position topology	Random positioning square grid
random seed	123,456
sky mote RAM	10 KB
sky mote flash	48 KB
sky mote clock speed	4 MHz
packet size	40 Bytes

Sky mote is equipped with 8 MHz MSP430 low power microcontroller, 10 KB RAM and 48 KB flash memory and also has 4 MHz of clock speed. The initial values of the simulation of each sensor node's output can, too, be saved to a file via menu options present in it, which will later be used for the machine learning methods. In the proposed model, the simulation was started and run for 10-hours for all scenarios. During the simulation, the "Mote output" window shows the outputs for each sky mote sensor nodes according to the timeline and displays the DODAG message details such as when the message is sent or received, and the node identification (ID) information. The Cooja simulation generates raw packet capture ".pcap" files, then these files can be converted into Comma Separated Values ".csv" files using Python. The proposed approach works independently, thus saving the nodes resources.

Energy consumption E (mW) value is the amount of energy used to send and receive data packets between motes (nodes) and calculated by (5)

$$E = \frac{energest\_value \times \mathrm{I} \times \mathrm{V}}{rtimer\_second \times runtime} \tag{5}$$

where the runtime is the time interval, *rtimer_second* is the number clock frequency, *energest_value* is the energy usage in two-time intervals (*runtime*) for a defined power state (e.g. CPU, LPM, transmit), I is current and V is voltage. So the total energy consumption, *TE,* is determined as follows.

$$\begin{aligned} TE = \frac{V}{t} \times ((I_{CPU} \times E_{CPU}) + (I_{LPM} \times E_{LPM}) \\ + (I_{Tx} \times E_{Tx}) + (I_{Rx} \times E_{Rx})) \end{aligned} \tag{6}$$

E_{CPU} is the total CPU energy consumption, E_{LPM} is the accumulated LPM energy consumption, E_{Tx} is the accumulated transmission energy consumption and E_{Rx} is the accumulated listen energy consumption in time t. The other parameters in (6) are defined in Table 3 and their values are taken from the Tmote Sky mote datasheet [42]. In addition, Contiki-Cooja simulation provides rimeaddr and transmitted values. Due to success rate of 99.96% of the proposed model, other features have not been included in this resource-constrained IoT environment.

TABLE 3. Energy consumption parameters for a constrained mote.

Parameter	Value
I_{CPU}, processing current consumption	1.8 mA
I_{LPM}, sleeping current consumption	0.0545 mA
I_{Tx}, transmission current consumption	19.5 mA
I_{Rx}, receiving current consumption	21.8 mA
V, voltage	3 V
runtime, time interval for measuring consumption values	1 s
rtimer_second, clock frequency	32768 ticks

CPU, LPM, Tx (transmit), Rx (receive) and TE values obtained from the motes during the simulation have been stored in datasets named "SSN1", "SSN2" and "SSN3", respectively based on the three scenarios created. The total numbers of normal and malicious motes/nodes are different in the scenarios.

In the simulation, new nodes join the network by broadcasting a "Hello" message with ID number and signal power, and announce their existence to their neighbors. Then all the other nodes update their routing table and send their own messages. In HF attack, malicious nodes frequently broadcast "Hello" messages by DIS packets looking like a neighbor and become the most available node for other nodes. So they cause their neighbors to spend their resources on processing wasteful packets [1]. Figure 6 shows the situations before and after HF attack in the simulation.

The average number of DIO, DIS and DAO message transmissions increases as the node's degree or rank increases.

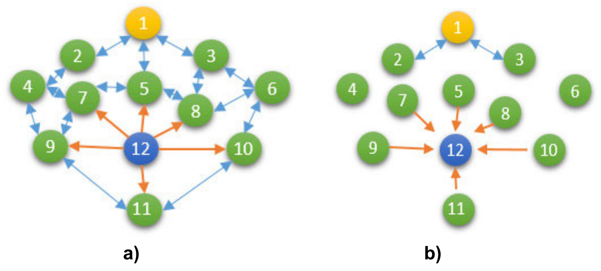

FIGURE 6. The situations a) before b) after HF attack.

As network size grows, transmission time increases due to the multiplicative effect of a greater hop count between the root and any other node. If the HF malicious node is in the first rank, for example, $40 \times 8 = 320$ bits of data will be transferred for a packet of 40 bytes. In addition, the bandwidth is equal to 10 Kbyte = 10240 bits. It is assumed that the value of h for a packet of 40 bytes (for sky mote) is 320 / 10240 = 0.031 s [43].

IV. PROPOSED MODEL

Attacks on routing protocols can cause great damage to resource-restricted networks, just like any other attack type. In this study, a model for detection and identification of attacks is proposed for a resource-constrained IoT environment. The overall architecture of the proposed model is illustrated in Fig. 7.

The architecture of the proposed model consists of three parts: network simulation, data preprocessing and attack detection. Firstly, the network is simulated by the Cooja simulator based on the scenario incorporating root, normal and malicious nodes. CPU, LPM, Tx, Rx and TE values of each node are calculated (see Section III). The message packets from the nodes are captured and filtered based on their contents. Secondly, all the data with relation to the result of the simulation are recorded in a ".csv" file. The data on the selected features (ID number, CPU, LPM, Tx, Rx, and TE) and operation time are stored in datasets named "SSNx", (an Excel file) depending on the scenario. Finally, the data set is normalized in the range of [0, 1] to improve the performance and is analyzed by GRU, Support Vector Machine (SVM), and LR machine learning methods. The output obtained indicates whether the node is normal or malicious depending on value range of Rx determined by statistical operations. Thus the packets of the attacking nodes will be dropped from the network when they are detected using the algorithm below. In the literature, drop and quarantine processes have been observed, so the drop process has been preferred. In this study, different combinations of features are also used to determine the most effective method in the detection of HF attacks.

In terms of using CPU, LPM, Tx, Rx and TE power calculations, and GRU deep learning method together, the proposed model will add novelty to the literature in the detection of HF attacks, which are one of the effective attack types in IoT.

The time complexity of the proposed method has been computed and presented as follows:

The proposed method	**# of times**
// R= Root (sink) node	1
// N= Other node	1
//N0=Neighbor Routing Table	1
START:	-
Set Root= R // broadcast the DIO message to establish the DODAG tree	1
Set Node= N // get DIO message	1
Set Node= N0 // nodes create their routing table by selecting their parent node	1
Set Node= N // send DAO message to R	1
Set Root= R // multicast the DAO-ACK to N	1
Set NewNode= N // send DIS packets to join the DODAG tree	1
DO:	-
Set R = N // sends DIO to N for their CPU, LPM, Tx, and Rx values	n-1
Set N = R // sends DAO to R for their CPU, LPM, Tx, Rx, and TE values	n-1
// ST = Simulation Time	-
while (ST<=36000s)	n
return (R)	1
END	-
START:	-
Set Feature Preprocessing (ID, CPU, LPM, Tx, Rx, and TE)	1
fn = combination number of the features (CPU, LPM, Tx, Rx, and TE)	1
DO	-
Set NewDataset= Feature Selection (ID, CPU, LPM, Tx, Rx, and TE) for ft step	n-1
GRU (Gated Recurrent Algorithm) [44]	n
while (ft <= fn) // tried to all combination of the features	n
return (evaluation metrics)	1
value range of Rx determined by statistical operations	1
END	-
START	-
DO	-
If max(Rx) > Rx > min(Rx)	n-1
Normal Nodes (makes an "N")	1
Else Malicious node detected and DROP Packet (make as "M" Hello Flood (HF) node)	n-1
// SD= Size of Dataset	-
while (t <= SD)	n
return (R)	1
END	-

The time complexity of the GRU is $O(N_I(H^2 + HN_o))$, where N_I is the number of features in inputs, H is the number of hidden units, and N_o is the number of outputs [44]. The time complexity of operations performed in Cooja simulation has been computed as $O(n)$. It is similar to [45] but higher accuracy value is obtained in this study. While attacks can be detected with 99.5% accuracy in [1], it is achieved using a single feature (Rx) and thus the computational load of the algorithm is also reduced. The solutions for HF attacks have been proposed in [19] but these are mainly cryptographic

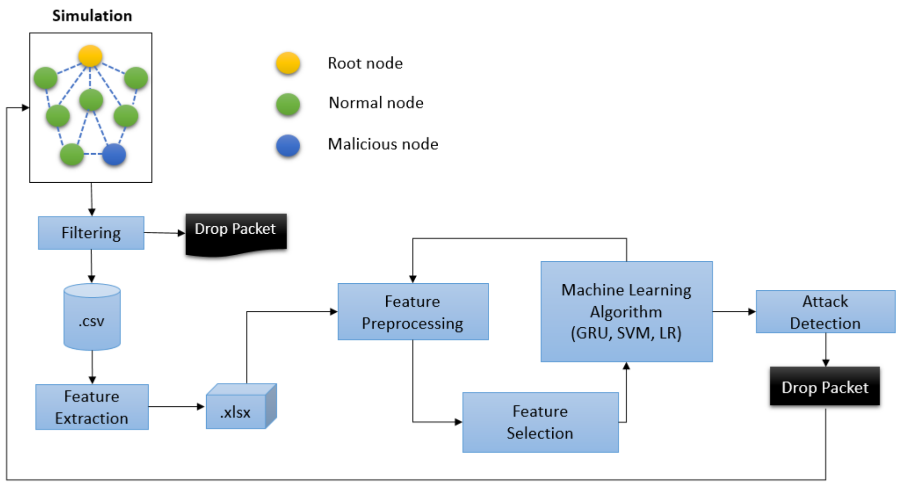

FIGURE 7. The architecture of HF attack detection and prevention model.

```
HF Attack Prevention Algorithm
Start
1st Step: Root node "R" broadcast the DIO
2nd Step: Other nodes "N" gets the message DIO
3rd Step: Other nodes "N" sends DAO to the root node "R"
4th Step: Root node "R" multicast the DAO-ACK to node "N"
    If Can join the DODAG
    Else Cannot join the DODAG
5th Step: DODAG will be constructed by repeating the steps 1-4
6th Step: Root node "R" sends the DIO (to receive the CPU, LPM, Tx, Rx, and TE values)
7th Step: Node "N" sends DAO (CPU, LPM, Tx, Rx, and TE values)
8th Step: Node "R" sends DAO-ACK
If $\max(Rx) > Rx > \min(Rx)$
    Normal Nodes (makes an "N")
Else Malicious node detected and DROP Packet (make as "M" Hello Flood (HF) node)
End
```

and cause heavy computational complexity. So they are not suitable for constraint devices in IoT. As can be seen in Table 1, GRU deep learning method proposed in the study yields better performance values than that of the other studies carried out on HF attacks.

In this study, there will be no resource and energy constraints on the nodes as the simulation and deep learning operations are performed in different servers. The average number of DIO message transmissions increases as the node's degree or rank increases. As network size grows, transmission time increases due to the multiplicative effect of a greater hop count between the root and any other node.

There is no modification in DIO, DIS or DAO messages, Contiki has a software-based power profiling system that mainly measures the time during which various components are activated. Some features belonging to each node (CPU, LPM, Tx and Rx) can be requested from the server via DIO message within the network. These features are used to determine TE value of each node in Contiki OS Cooja Simulator. The feature values of the sensor nodes for power consumptions will be collected and printed for every 10 seconds of the Contiki clock (clock_second $\times$ 10). The RPL incurs control overheads (DIO, DAO, and DIS) during the DODAG construction. Each node transmits DIO messages using the trickle timer based on the network status and the frequency of DIO messages also depends on the network stability [44].

In this study, GRU method has been applied manually after the nodes have been created in the Cooja simulator. The obtained results have been optimized and the intrusion detection threshold has been determined through the machine learning methods (GRU, SVM, LR) used in the study. In the proposed method, the simulation and deep learning methods do not create high computational loads as they run separately.

V. RESULTS AND DISCUSSION

In this study, dealing with the types of HF attacks, the power states of the nodes have been examined before and after the attack. Time-dependent energy consumption values of non-malicious (normal) and malicious nodes are presented graphically in Fig. 8 and Fig. 9, respectively.

As can be seen Fig. 8, the changes in LPM and Rx occurs at very large intervals, but in CPU and Tx at smaller intervals. Fig. 9 shows that the energy usage of the malicious node in the LPM varies at a certain level and especially in Rx, this is quite low compared to the non-attack situation. CPU energy usage is quite low in both graphs, but the change in malicious node is clearly seen.

Packet Delivery Rate (PDR) and delay have also been considered to evaluate the network performance. Networks with different number (2-4-8 and 16) of HF malicious nodes

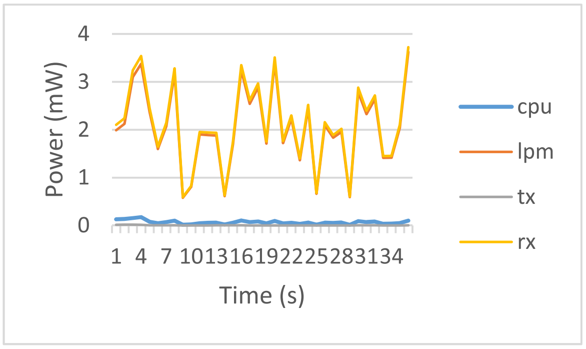

FIGURE 8. Energy consumption of non-malicious nodes.

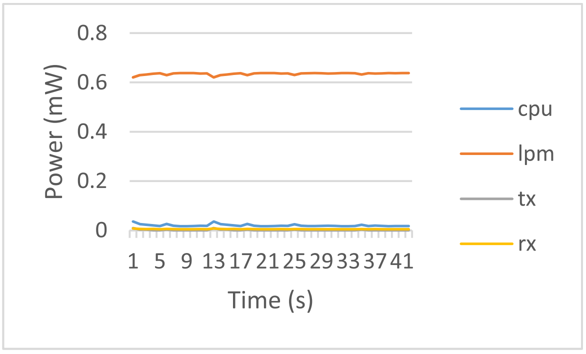

FIGURE 9. Energy consumption of malicious node.

have been created and then simulated 900000 ms. The changes of PDR and delay based on the number of malicious nodes have been computed and illustrated in Fig. 10 and Fig. 11, respectively.

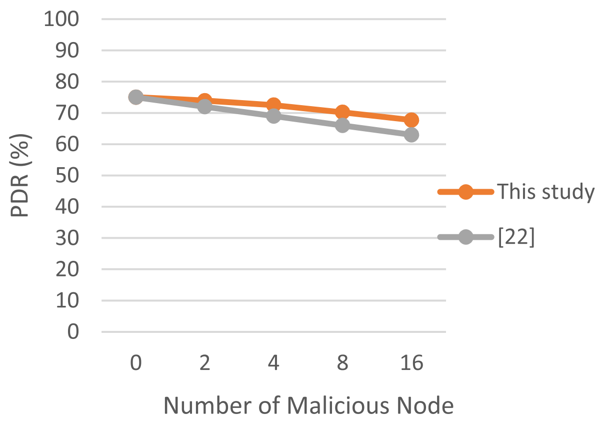

FIGURE 10. The changing of PDR with the number of malicious node.

It is clear from the figures that when the number of HF malicious nodes is increased, the performance of the network decreases (e.g. as increase in the time for all packets to reach their destination and in the number of lost packets). In addition, the delay is less in the proposed study, although there are more packets sent compared with that of Khosravi *et al.* [22]. This study estimating better results can be used for overcoming HF attacks.

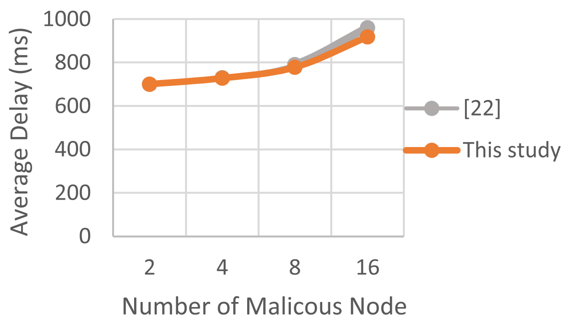

FIGURE 11. Average end-to-end delay with number of malicious node.

In this study, GRU neural networks are used to accurately detect HF attacks in all scenarios. The dataset containing ID, CPU, LPM, Tx, Rx and TE data has been divided into 2 subsets for training and test of GRU networks: 80% for training set and 20% for testing. GRU network models have been developed using Keras in Python. Dropout value is chosen as 20% to prevent overfitting and tangent hyperbolic activation function is preferred. As a result of the experimental evaluations to determine the best network model and architecture for the GRU method used in attack detection, the values given in Table 4 are determined.

TABLE 4. Network model characteristics.

Characteristic	Value
test size	0.2
random state	42
input	30
hidden layer	30
output	10
dropout	20%
activation function	tanh
optimizer	Adam
loss	Mean-squared error
epoch	50
batch size	256

SSN1 dataset has been used to compare the performance of GRU, SVM, and LR machine learning methods and the size of the data set is 10519 x 6. GRU, SVM, and LR are run separately 50 epochs for combinations of different features, the accuracy rates changing by used features are presented in Table 5. The best results are obtained with GRU deep learning method for each combination and it has also performed better than SVM and LR in most cases. When the accuracy rates obtained using one feature are compared, it is seen that the highest accuracy value is obtained with Rx and the worst performances for all methods are the ones the Tx is used only.

As can be seen in Table 5, the highest accuracy (99.95%) has achieved by GRU using CPU, Tx, Rx and TE values. The accuracy rate has decreased by 0.15 with the addition of LPM value (CPU, LPM, Tx, Rx, TE). Accuracy rates

TABLE 5. Performance comparison for different features.

Feature(s)	Accuracy (%)		
	GRU	SVM	LR
CPU, LPM, Tx, Rx, TE	99.80	99.00	98.95
CPU, LPM, Tx, TE	97.96	94.62	93.72
CPU, LPM, Rx, TE	99.51	99.22	98.95
CPU, Tx, Rx, TE	99.95	99.85	98.81
LPM, Tx, Rx, TE	99.26	98.10	97.95
CPU, LPM, Tx, Rx	99.9	99.10	98.90
Tx, Rx, TE	99.81	99.57	98.81
CPU, Rx, TE	99.22	98.85	98.81
CPU, LPM, TE	94.62	92.04	91.72
CPU, Tx, TE	96.96	93.00	91.53
CPU, LPM, Tx	95.15	94.62	93.15
LPM, Tx, Rx	99.81	98.86	98.81
CPU, Tx, Rx	99.81	99.85	98.76
Rx, TE	99.86	99.57	98.81
CPU, TE	95.24	93.05	91.58
CPU, LPM	94.91	94.62	93.05
LPM, Tx	96.05	94.67	93.05
Tx, Rx	99.81	99.57	98.47
Tx	79.55	56.96	57.96
Rx	99.52	99.57	98.47
TE	85.69	83.40	83.59
CPU	94.72	92.86	92.10
LPM	96.05	94.57	93.00

obtained using the (Tx, Rx), (LPM, Tx, Rx), and (CPU,Tx, Rx) combinations are equal to %99.81. When the CPU, LPM, Tx and Rx values are fed into the network as input, an increase of approximately 0.09 has occurred in the accuracy.

The use of Rx values has provided high accuracy value, and Tx and LPM values have increased the performance less compared to Rx, respectively. 99.52% success rate has been achieved only with the Rx value rather than monitoring parameters (CPU, LPM, Tx, Rx and TE) for the nodes. Therefore, it can be said that Rx feature provides strong superiority over other ones in order to drop the packets after detecting node. The minimum and maximum Rx values obtained from the dataset have been used to detect HF attacks and malicious nodes as given in the algorithm (see Section IV).

The SSN1, SSN2 and SSN3 datasets created as a result of simulations made according to different scenarios are given as inputs to GRU, SVM and LR. Then the obtained results using ACC (Accuracy), MSE (Mean Squared Error), MAE (Mean Absolute Error) and RMSE (Root Mean Square Error) evaluation metrics are presented comparatively in Table 6.

As seen, the accuracy values have increased with the increase in the number of nodes for both of the feature sets. The best reported result is 99.96% and the differences in the number of nodes do not cause major changes in detecting malicious nodes. The proposed method for HF detection has successful compared to the existing and related studies (see Table 1) using GRU deep learning method regardless of the number of nodes.

MSE, MAE and RMSE values are less in most cases, by about 0.01, 0.05 and 0.05 in three scenarios with two feature sets, respectively. Based on a rule of thumb, it can be said that the lower the value, the better the performance of

TABLE 6. Performance of GRU in different scenarios.

Scenario	Metric	Feature set	
		CPU,LPM,Tx,Rx,TE	CPU,LPM,Rx,TE
1	ACC (%)	99.80	99.51
	MSE	0.0082	0.0141
	MAE	0.0316	0.0418
	RMSE	0.0455	0.0434
2	ACC (%)	99.93	99.80
	MSE	0.0065	0.0081
	MAE	0.0361	0.0461
	RMSE	0.0411	0.0423
3	ACC (%)	99.96	99.90
	MSE	0.0075	0.0041
	MAE	0.0364	0.0159
	RMSE	0.0284	0.0168

the model would be. The best result of ACC, MSE, MAE and RMSE is reported in 3rd scenario. This also shows that the model can accurately predict the attacks.

VI. CONCLUSION

Deep learning methods yield the most successful results in IoT routing attack detection systems as in many areas. In this study, GRU neural networks-based deep learning are preferred due to their simple structure (e.g. adding new gates with fewer codes), ability to learn the model faster with less data compared to LSTM. Different combinations of CPU, LPM, Tx, Rx and TE power states used for the training of GRU in order to detect HF attacks. The results obtained using GRU, SVM and LR methods are given comparatively in Table 5. Different scenarios have been created and the performance of GRU is compared for different number of the normal/malicious nodes. As can be seen in Table 5 and Table 6, the highest accuracy rate of this study is 99.96% and GRU is more successful in terms of delay and PDR for attack detection (see Fig. 10 and Fig. 11). Certainly, it is predicted that there could be a scalability problem when the number of nodes is increased heavily. This is identified as the subject of a further study. In addition, the proposed model may also give new insights and trigger new attempts on detection and prevention of routing attacks in IoT environments. GRU based deep learning method can be used with high performance to detect and prevent RPL attacks in IoT. In the future studies, the method proposed is planned to be used in the detection and prevention of other attack types.

REFERENCES

[1] F. Y. Yavuz, D. Ünal, and E. Gül, "Deep learning for detection of routing attacks in the Internet of Things," *Int. J. Comput. Intell. Syst.*, vol. 12, no. 1, pp. 39–58, Nov. 2018.
[2] E. Kfoury, J. Saab, P. Younes, and R. Achkar, "A self organizing map intrusion detection system for RPL protocol attacks," *Int. J. Interdiscipl. Telecommun. Netw.*, vol. 11, no. 1, pp. 30–43, Jan. 2019.
[3] P. Pongle and G. Chavan, "Real time intrusion and wormhole attack detection in Internet of Things," *Int. J. Comput. Appl.*, vol. 121, no. 9, pp. 1–9, Jul. 2015.
[4] A. Mayzaud, R. Badonnel, and I. Chrisment, "A taxonomy of attacks in RPL-based Internet of Things," *Int. J. Netw. Secur.*, vol. 18, no. 3, pp. 459–473, May 2016.
[5] X. Yuan, C. Li, and X. Li, "DeepDefense: Identifying DDoS attack via deep learning," in *Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP)*, May 2017, pp. 1–8.

[6] G. Thamilarasu and S. Chawla, ''Towards deep-learning-driven intrusion detection for the Internet of Things,'' *Sensors*, vol. 19, no. 9, p. 1977, Apr. 2019.

[7] R. Khan, S. U. Khan, R. Zaheer, and S. Khan, ''Future Internet: The Internet of Things architecture, possible applications and key challenges,'' in *Proc. 10th Int. Conf. Frontiers Inf. Technol.*, Dec. 2012, pp. 257–260.

[8] A. Riahi Sfar, E. Natalizio, Y. Challal, and Z. Chtourou, ''A roadmap for security challenges in the Internet of Things,'' *Digit. Commun. Netw.*, vol. 4, no. 2, pp. 118–137, Apr. 2018.

[9] H. Xie, Z. Yan, Z. Yao, and M. Atiquzzaman, ''Data collection for security measurement in wireless sensor networks: A survey,'' *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2205–2224, Apr. 2019.

[10] J. Kim, J. Lee, J. Kim, and J. Yun, ''M2M service platforms: Survey, issues, and enabling technologies,'' *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 61–76, 1st Quart., 2014.

[11] E. J. Hui and P. Thubert, *Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks*, document RFC 6282, Internet Requests for Comments, 2011, pp. 1–24.

[12] *IEEE Standard for Local and Metropolitan Area Networks—Part 15.4: Low-Rate Wireless Personal Area Networks (WPANs)*, Standard IEEE Std 802.15.4-2011, M. Standards Committee of the IEEE Computer Society 2006.

[13] L. Bartolozzi, T. Pecorella, and R. Fantacci, ''Ns-3 RPL module: IPv6 routing protocol for low power and lossy networks,'' in *Proc. 5th Int. Conf. Simul. Tools Techn.*, 2012, pp. 359–366.

[14] A. L. Colina, A. Vives, A. Bagula, M. Zennaro, and E. Pietrosemoli. (2015). *IoT in 5 Days*. BookSheet. [Online]. Available: http://wireless.ictp.it/school_2015/book/book.pdf

[15] A. U. Gawade and N. Shekokar, ''Lightweight secure RPL: A need in IoT,'' in *Proc. Int. Conf. Inf. Technol.*, Dec. 2017, pp. 214–219, doi: 10.1109/ICIT.2017.31

[16] R. Singh, J. Singh, and R. Singh, ''Hello flood attack countermeasures in wireless sensor networks,'' *Int. J. Comput. Sci. Mobile Appl.*, vol. 4, no. 5, pp. 1–9, May 2016.

[17] V. PalSingh, A. S. A. Ukey, and S. Jain, ''Signal strength based hello flood attack detection and prevention in wireless sensor networks,'' *Int. J. Comput. Appl.*, vol. 62, no. 15, pp. 1–6, Jan. 2013.

[18] T. Sherasiya, H. Upadhyay, and H. B. Patel, ''A survey: Intrusion detection system for Internet of Things,'' *Int. J. Comput. Sci. Eng.*, vol. 5, no. 2, pp. 91–98, 2016. [Online]. Available: http://www.iaset.us/view_archives.php?year=2016&id=14&jtype=2&page=2

[19] V. P. Singh, S. Jain, and J. Singhai, ''Hello flood attack and its countermeasures in wireless sensor networks,'' *Int. J. Comput. Sci.*, vol. 7, no. 11, p. 23, May 2010.

[20] A. Verma and V. Ranga, ''Addressing flooding attacks in IPv6-based low power and lossy networks,'' in *Proc. IEEE Region Conf. (TENCON)*, Oct. 2019, pp. 552–557, doi: 10.1109/TENCON.2019.8929409.

[21] H. I. Ahmed, A. A. Nasr, S. Abdel-Mageid, and H. K. Aslan, ''A survey of IoT security threats and defenses,'' *Int. J. Adv. Comput. Res.*, vol. 9, no. 45, pp. 325–350, Oct. 2019. [Online]. Available: https://search.proquest.com/docview/2307941033?accountid=25074, doi: 10.19101/IJACR.2019.940088.

[22] H. Khosravi, I. Alzahra University, R. Azmi, and M. Sharghi, ''Adaptive detection of hello flood attack in wireless sensor networks,'' *Int. J. Future Comput. Commun.*, vol. 5, no. 2, pp. 99–103, 2016.

[23] T. Sherasiya and H. Upadhyay, ''Intrusion detection system for Internet of Things,'' *Int. J. Advance Res. Innov. Ideas Edu.*, vol. 2, no. 3, pp. 2244–2349, 2016.

[24] P. Kasinathan, C. Pastrone, M. A. Spirito, and M. Vinkovits, ''Denial-of-service detection in 6LoWPAN based Internet of Things,'' in *Proc. IEEE 9th Int. Conf. Wireless Mobile Comput., Netw. Commun. (WiMob)*, Oct. 2013, pp. 600–607.

[25] S. Shamshirband, N. B. Anuar, M. L. M. Kiah, V. A. Rohani, D. Petkovic, S. Misra, and A. N. Khan, ''Co-FAIS: Cooperative fuzzy artificial immune system for detecting intrusion in wireless sensor networks,'' *J. Netw. Comput. Appl.*, vol. 42, pp. 102–117, Jun. 2014, doi: 10.1016/j.jnca.2014.03.012.

[26] S. Shamshirband, A. Amini, N. B. Anuar, M. L. Mat Kiah, Y. W. Teh, and S. Furnell, ''D-FICCA: A density-based fuzzy imperialist competitive clustering algorithm for intrusion detection in wireless sensor networks,'' *Measurement*, vol. 55, pp. 212–226, Sep. 2014.

[27] I. N. Aizenberg, N. N. Aizenberg, and J. Vandewalle, ''Multiple-Valued threshold logic and multi-valued neurons,'' in *Multi-Valued Universal Binary Neurons*. Boston, MA, USA: Springer, 2000, pp. 25–80.

[28] O. Brun, Y. Yin, J. Augusto-Gonzalez, M. Ramos, and E. Gelenbe, ''IoT attack detection with deep learning,'' in *Proc. ISCIS Secur. Workshop*, Londra, U.K., Feb. 2018, Art. no. 02062091.

[29] A. Seker, B. Diri, and H. Balik, ''A review about deep learning methods and applications,'' *Gazi J. Eng. Sci.*, vol. 3, no. 3, pp. 47–64, 2017.

[30] M. A. Hamid, M. Rashid, and C. S. Hong, ''Routing security in sensor network: Hello flood attack and defense,'' in *Proc. IEEE ICNEWS*, Jan. 2006, pp. 2–4.

[31] C. Karlof and D. Wagner, ''Secure routing in wireless sensor networks: Attacks and countermeasures,'' *Ad Hoc Netw.*, vol. 1, nos. 2–3, pp. 293–315, Sep. 2003.

[32] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, ''SPINS: Security protocols for sensor networks,'' *Wireless Netw.*, vol. 8, no. 5, pp. 521–534, Sep. 2002.

[33] *Keras Documentation: About Keras*, Accessed: Jan. 11, 2020. [Online]. Available: https://keras.io/about/

[34] Y. Bengio, P. Simard, and P. Frasconi, ''Learning long-term dependencies with gradient descent is difficult,'' *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994, doi: 10.1109/72.279181.

[35] S. Hochreiter and J. Schmidhuber, ''Long short-term memory,'' *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[36] R. Fu, Z. Zhang, and L. Li, ''Using LSTM and GRU neural network methods for traffic flow prediction,'' in *Proc. 31st Youth Academic Annu. Conf. Chin. Assoc. Autom. (YAC)*, Nov. 2016, pp. 324–328.

[37] M. Nguyen. (2018).*Illustrated Guide to LSTM's and GRU's: A Step by Step Explanation*. Accessed: Jan. 10, 2020. [Online]. Available: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

[38] G. Drakos. (2019). *What is a Recurrent Neural Networks (RNNS) and Gated Recurrent Unit GRUS*. Accessed: Jan. 10, 2020. [Online]. Available: https://medium.com/@george.drakos62/what-is-a-recurrent-nns-and-gated-recurrent-unit-grus-ea71d2a05a69

[39] C. C. Chatterjee. (2019). *Implementation of RNN, LSTM, and GRU*. Accessed: Jan. 10, 2020. [Online]. Available: https://towardsdatascience.com/implementation-of-rnn-lstm-and-gru-a4250bf6c090

[40] S. Kostadinov. (2017). *Understanding GRU Networks*. Accessed: Jan. 13, 2020. [Online]. Available: https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be

[41] Contiki. (2015). *Contiki: The Open Source Operating System for the Internet of Things*. Accessed: Jan. 15, 2020 [Online]. Available: http://www.contiki-os.org/

[42] S. Raza, L. Wallgren, and T. Voigt, ''SVELTE: Real-time intrusion detection in the Internet of Things,'' *Ad Hoc Netw.*, vol. 11, no. 8, pp. 2661–2674, Nov. 2013.

[43] S. Toklu and O. Ayhan Erdem, ''BSC-MAC: Energy efficiency in wireless sensor networks with base station control,'' *Comput. Netw.*, vol. 59, pp. 91–100, Feb. 2014.

[44] A. Musaddiq, Y. B. Zikria, Zulqarnain, and S. W. Kim, ''Routing protocol for low-power and lossy networks for heterogeneous traffic network,'' *EURASIP J. Wireless Commun. Netw.*, vol. 2020, no. 1, p. 21, Dec. 2020.

[45] N. Müller, P. Debus, D. Kowatsch, and K. Böttinger, ''Distributed anomaly detection of single mote attacks in RPL networks,'' in *Proc. 16th Int. Joint Conf. e-Bus. Telecommun.*, vol. 2, 2019, pp. 378–385, doi: 10.5220/0007836003780385.

SEMIH CAKIR was born in Zonguldak, Turkey, in 1986. He received the B.S. degree (Hons.) in computer and instructional teacher education from Anadolu University, Eskisehir, in 2010, and the M.S. degree in computer engineering from Bilecik Seyh Edebali University, Bilecik, Turkey, in 2012. He is currently pursuing the Ph.D. degree in electrical, electronics, and computer engineering with Düzce University, Düzce, Turkey. From 2012 to 2017, he worked as an Instructor at Amasya University, Amasya, Turkey. Since 2017, he has been an Instructor and an Assistant Director of the Kdz. Eregli Vocational School, Zonguldak Bulent Ecevit University. His research interests include the Internet of Things, cyber security, computer networks, deep learning, cloud computing, and programming languages at the undergraduate degree.

SINAN TOKLU was born in Adana, Turkey, in 1979. He received the B.Sc. degree in computer engineering from Eastern Mediterranean University in 2004, and the M.Sc. degree in computer engineering and the Ph.D. degree in electrical–electronics education from Gazi University, Ankara, Turkey, in 2007 and 2013, respectively. From 2007 to 2010, he worked as Research Assistant at Gazi University. From 2010 to 2014, he worked as a Software Developer at TEİAŞ, Turkey. Since 2014, he has been an Assistant Professor with the Department of Computer Engineering, Düzce University. His research interests include the Internet of Things, WSN, deep learning, and smart city.

NESIBE YALCIN was born in Yozgat, Turkey, in 1987. She received the B.S. (Hons.) and M.S. degrees in computer engineering from Selcuk University, Konya, in 2009 and 2012, respectively, and the Ph.D. degree in computer and information engineering from Sakarya University, Turkey, in 2017. From 2011 to 2017, she worked as a Research Assistant at Bilecik Seyh Edebali University, Turkey. Since 2018, she has been an Assistant Professor and the Co-Chair of the Department of Computer Engineering, Bartın University. Her teaching areas are, but not limited to, machine learning, artificial neural networks, data mining, and heuristic optimization algorithms at both under and postgraduate levels. Her research interests include the Internet of Things, artificial intelligence applications, and mathematical modeling and simulation.