Journal of Computational Science 62 (2022) 101718

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

A node pairing approach to secure the Internet of Things using machine learning

Usman Ahmad

School of Computer Science and Technology, Beijing Institute of Technology, China

ARTICLE INFO

Keywords:
Internet of Things (IoT)
Security attacks
Machine learning
Sensor nodes
Wireless Sensor Networks (WSNs)

ABSTRACT

Although the reception of Internet of Things (IoT) services increased efficiency and allowed companies to automate the production through remote and intelligent control, it also increases the risk of potential threats and security attacks. The attackers can compromise the security of the IoT node and join it to a botnet, manipulate sensitive data or take control of a full network. Securing the IoT nodes from security attacks remains a challenge, especially under the multiple attack models where several attacks can be launched simultaneously. The objective of this paper is to prevent a type of multiple attacks that are the foundation to manipulate sensor nodes' data that are: clone attack, tamper attack, false data injection, node malfunctioning, and physical layer attack. A solution is proposed to secure the IoT nodes against these attacks. The proposed solution is based on a novel sensor pairing approach to evaluate the trustworthiness of IoT nodes' sensor data using multiple machine learning models. In particular, we propose a threat model and the pairing algorithm to pair each sensor node with the sensor node located in the neighbor. We compare the performance of three machine learning models, which are decision tree, Support Vector Machine (SVM), and k-Nearest Neighbors (KNN) by using two publically available real-world datasets in terms of attack detection accuracy, training time, and testing time. Our proposed system achieved up to 100% accuracy rate and nearly twice faster training and testing time as compared to the existing solutions. We provide the memory and energy consumption analysis to show that the proposed method has lower communication and memory overhead on IoT nodes than previous solutions against IoT security attacks. Our dataset just contains the data of a pair of sensors and is reduced up to eight times smaller in size than the actual dataset.

1. Introduction

The Internet of Things (IoT) is one of the key research areas in both industry and education. The IoT has advanced to comprise numerous next-generation network paradigms, including Wireless Sensor Networks (WSNs), device-to-device communication, cognitive radio, and other technologies. The WSN is the collection of millions of nodes that are distributed in the different regions of interest to form a giant network to collect and transfer data to analysis. There can be many different types of WSNs incorporated to form an IoT system. The application areas of the IoT are manifold since it is adaptable and adjustable to almost any technology which is accomplished by providing relevant information about events and activities about the environmental conditions that need to be analyzed and remotely controlled. IoT technology has gained momentum and is now shaping our future [1]. Nowadays, many organizations are adopting IoT technology to improve and automate their processes. There are several surprising practical application areas of the IoT.

Security of IoT is an important issue. In [2], the author categorized the IoT security attacks in 31 different types. Some of the major attacks discussed by the author are: node malfunctioning, tempering, replication, Sybil (multiple identities to neighbor nodes), and node destruction/outage. These attacks further result in spoofing, sensor overwhelming/treats, eavesdropping, traffic analysis, black/gray/sink/wormhole attacks, de-synchronization/unfairness, and routing attacks. Moreover, multiple request attacks result in DDoS attacks, denial of sleep, jamming/flooding, and collusion/exhaustion. The Industry of security spending millions of US dollars to secure the IoT and its graph is rising every year [3]. The existing solutions to secure the WSNs can be categorized as encryption, allow limited requests, anomaly detection, identification/verification techniques, bandwidth, and speed test-based solutions, honeypots, multipath routing, and physical (hardware) security. Most of the solutions are proposed on the network layer. Machine learning [4,5] and deep learning [6] also has been extensively accepted as a mechanism for attack detection.

E-mail address: 3820170112@bit.edu.cn.

https://doi.org/10.1016/j.jocs.2022.101718
Received 22 November 2021; Received in revised form 19 May 2022; Accepted 19 May 2022
Available online 1 June 2022

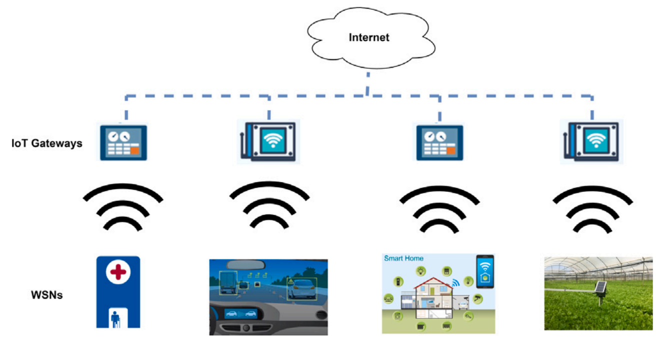

Fig. 1. IoT and WSNs applications.

1.1. *IoT architecture*

The IoT applications are widely adopted for monitoring and collecting data about physical environmental conditions. WSNs are the foundation of IoT. The architecture of WSNs is based on wireless sensor nodes which are used to gather intelligible data like water flow, heat, humidity, position, vibration, etc., and share them wirelessly with the base station. A base station or sink is like an interface between WSNs and internet/users. Users can get required data from the WSNs by injecting queries and gathering results from the base station. Typically the WSNs comprise hundreds of thousands of nodes, which are interconnected and communicate with each other using radio signals. The working approach of the sensor nodes may be either event-driven or continuous. Wireless sensor devices are also equipped with actuators to take action in certain circumstances. The count of wireless nodes is predicted to range around 100 billion by the end of 2025 [7], as it is the foundation as regards IoT and Industrial IoT (IIoT) applications. The IoT application areas have grown fast in numerous fields including health care, smart homes and cities, industrial control, smart vehicles, environmental monitoring, and many others as shown in Fig. 1.

1.2. *Research challenges and motivation*

Although the acceptance of IoT services allowed companies to automate production, it also increases the risk of potential threats and security attacks. The attackers can compromise the security of the IoT node and manipulate sensitive data. Securing the IoT nodes from data manipulation attacks remains a challenge. A sensor node collects and transmits data by connecting to another sensor node or a gateway. Multiple attacks can be performed to manipulate sensor data. Some of the common attacks incorporated in sensor data manipulation are clone attacks, tamper attacks, false data injection, node malfunctioning, and physical layer attacks.

An attacker can physically grasp or inject false data to the sensor nodes, temper it and install various replicated sensor nodes. These compromised have legitimate access to the network (ID, code, cryptographic information, etc.), so can equally contribute to the network actions as the authentic nodes and transmit malicious data. The attacker can also inject false/malicious data about the environment to the sensor node in order to change the working of the actuator or even affect the full network. So, it is compulsory to efficiently detect such types of attacks in order to limit the damage and ensure the standard working of the system.

Another research challenge is that the wireless sensor nodes are typically lower in price and smaller in size, so have limited memory and battery capacity. Thus, the system designed to detect such types of data manipulation attacks should not only assure high performance but also take into count the memory and energy constraints of sensors. There are various existing research efforts for attack detection in sensor networks, e.g., witness-based, key schemes to protect sensors from being compromised. Various solutions are discussed in Section 2.

1.3. *Contribution*

Our proposed solution not only successfully detects the data manipulation attacks but also does not have memory or network overhead on sensor nodes. We select to pair the sensor with the sensor located in the neighbor with the shortest path distance to confirm the legitimacy of sensor node data. In this paper, we provide two case studies to reflect the effectiveness of pairing and to evaluate our solution. Data perceived from the environment by each sensor node in the pair verifies the data of its paired sensor node. The selection of sensor nodes to be paired can be based on the location, sensitivity, sleep time, location, and movability of sensor nodes. We have also proposed a sensor pairing algorithm that pairs the selected sensor nodes located in the neighbor.

When the sensor node detects an event (temperature, flow, etc.), the event is conveyed to one of the base stations which then performs the appropriate action. The base station in our model is called the queen node. When the event is transmitted from sensor node 1 in a pair to the queen node, the queen node analyzes this event based on the event information gotten from sensor node 2 of the pair and vice versa. Queen node employs machine learning techniques to identify whether the data of the sensor node is normal or malicious. The previous solutions have memory overhead on sensor nodes and communication overhead on the network while our proposed solution does not have network traffic overhead or the need for sensors storage. The simplicity and independence of our proposed model make it universally adaptable. The contribution of this paper is as follows:

- We present a threat model where sensor nodes data is manipulated and transmitted to the base station in order to change the working of the actuator.
- We propose a novel paring structure to confirm the legitimacy of sensor nodes. We also propose the sensor node pairing algorithm.
- We provide the two case studies in different application areas of IoT to evaluate the performance of our model.
- We apply and compare the performance of decision tree (specifically CART), Support Vector Machine (SVM), and k-Nearest Neighbors (KNN) models by using two publically available real-world datasets in terms of attack detection accuracy, recall, precision, f-score, training time, and testing time.
- We performed the memory and energy consumption analysis in order to calculate the communication and memory overhead of the proposed solution.

The rest of the paper is structured as follows. Prior research work about IoT attack detection solutions is discussed in Section 2. Section 3 illustrates the threat model of security attack and our proposed solution based on the sensor pairing approach and machine learning models. In Section 4, we perform the experiment and demonstrate the results of two case studies to evaluate our model. Finally, conclusions are presented in Section 5.

2. Related work

Securing the IoT nodes remains a challenge and a vast research area, so the author has layer-wise distributed the IoT security attacks into 31 different categories, in which clone attack, tamper attack, false data injection, node malfunctioning, and physical layer attacks are the major categories [2]. Solutions against the WSNs security attacks can be summarized under 2 categories: centralized and distributed solutions. The authors discussed a distributed method for clone detection in [8]. In this scheme, a set of watchdog sensors is used to clone detection. In a recent study, the author takes benefit of the well-known SimHash method to protect users' location privacy by clustering the users for the IoT devices in the healthcare domain [9].

A famous witness-based distributed technique is proposed to detect the clones using the energy-efficient ring-based clone detection (ERCD) protocol [10]. ERCD protocol is a twofold process, which contains witness selection and legality confirmation. The author demonstrated an adversary model and two distributed protocols (HIP and HOP) to detect the clone attacks [11]. The problem regarding the designing of the sinkhole detection method in IoT is addressed in [12]. The proposed method is a sinkhole detection mechanism based on the routing protocol for low power and lossy networks (RPL) routing protocol.

A clone detection scheme is proposed for sensor networks in [13]. This is a distributed solution called Random Walk with Network Division (RWND) in static WSNs which is based on a witness framework and verifies the legitimacy of the sensor node. A double-track approach is proposed in [14] to detect the clones for the RFID base system by developing a double-track inspection on the consistency of related events. In [15], the author presents a distributed clone detection method called low-storage clone detection (LSCD) for WSNs. They design a detection route based on witness nodes. In the proposed solution, the clone is detected in a non-hotspot area which improves the energy efficiency and network lifetime. A distributed framework for clone detection called random walk with network division (RAND) is proposed and presented results to validate the impact of network size and the number of selected witnesses in [16].

An author demonstrated in [17], how machine learning is used to detect possible vulnerability exploits and exposed ten attack vectors. An innovative model is proposed to detect botnet-based DDoS attacks in IoT using machine learning [18]. Another machine learning-based solution is proposed [19] in employing software defined networking (SDN) to learn from the behavior of devices to detect the attacks against IoT. Machine learning is also used for malware detection. The author demonstrated a case study in which he detected the malware on an Android device [20]. In [21], the author applied LSTM recurrent neural network in healthcare to secure the body sensor network. Another method is proposed in [22], which is a mixture of machine learning approaches to detect relay attacks on smart vehicles. The author presents a model based on machine learning to detect the clones by extracting the features from the abstract syntax trees and program dependency graphs [23]. The author exploits the results of some deep learning models to find the clones using vector similarity measure, which calculates the similarity indicator between two code snippets [24]. In [25], the author proposed a scalable framework, called Clone-Hunter using binary code clone detection. The results of the paper show that the proposed framework can detect redundant bound checks faster than pure binary symbolic execution. A solution proposed in [26] that can cluster similar clones using a machine learning approach. The author also presented the NiCad clone detection system based on the Java class library which illustrates that their approach is effectively detecting the clones.

The authors proposed the algorithms for monitoring and controlling the IoT-enabled vehicle under security attacks in [27]. In [28], the author proposed an autonomous resource provisioning method for IoT applications combined with fog computing. The author proposed an autonomic resource provisioning technique known as a 3-dimensional technique to provide providence and flexibility features [29]. In [30], the author proposed an estimation algorithm to support the contribution of IoT networks in smart grid systems.

In summary, a common solution discussed in this section against node attacks is the base station to collect the information of the neighbors from each sensor and monitor the network if there is a compromised node. This model results in a high communication overhead. In the conventional witness-based technique, typically, a combination of sensors is nominated, which are called witnesses, to legitimate the sensor nodes in the network. The identity and the location information of the source sensor node are shared with witnesses for witness selection. When a node in the network needs to send data, firstly, it transmits the request to the witnesses for validity confirmation, and witnesses verify or generate an attack detection alarm when the sensor is unsuccessful in the certification. This technique has memory and network overhead on sensor nodes. Related research on IoT security solutions is shown in Table 1.

3. Methodology

In this section, we describe the threat model and proposed methodology. In our threat model, we considered all types of security attacks that are the foundation to manipulate the sensor's data. Our proposed solution consists of two major parts: a sensor pairing approach and the machine learning algorithm-based analysis. The sensor pairing algorithm pairs each sensor node with the sensor node located in the neighbor. We apply and compare the performance of three well-known machine learning algorithms which are: the CART, SVM, and KNN models.

3.1. Threat model

Since the sensor nodes are not inviolable, they can be attacked and the attacker can read and manipulate the sensor data. Moreover, the attacker can make replicas of the nodes at random locations in the network, which can transmit malicious data. We focused on the main issue of data alteration which occurs due to multiple security attacks, listed in this section. Data alteration is a direct attack on data authenticity and integrity. In this scenario, the sensor nodes transmit malicious data about the environment to the base station.

In this model, we consider an IoT network containing multiple WSNs with multiple base stations and a large number of wireless sensor nodes that are randomly distributed in the network. A base station is the origin of the system controller. In the densely deployed WSNs, for each sensor node, there exist sensor nodes located in the neighbor. Moreover, the network can be extended by adding more base stations, where different base stations can communicate with its sensor nodes. Our proposed solution is effective, even the link-level security definite by deploying the powerful cryptography scheme is also compromised. Our solution proposed is against the sensor node data manipulation which occurs due to multiple security attacks as given below:

Clone attack. A clone attack is one of the most terrible attacks in which an attacker places replicas of compromised sensor nodes in several places in the network to divert the behavior of the network. The attackers have the ability to compromise a bag of sensor nodes placed at random locations in the network. These replicated sensor nodes can transmit malicious/altered data and affect the working of the network in several ways.

Node tampering attack. In node tampering, the attackers can temper the electronic board and its wiring or temper the memory contents of the sensor node and misuse the captured slave sensor node. These tempered nodes can transmit false data and launch many other attacks like DoS attacks etc.

False data injection. Attackers can inject false data into the network through compromised nodes. The compromised nodes mislead data by injecting false data during data-forwarding. This false data leads the base station and actuators to make false decisions.

Node malfunctioning. This attack may occur because of different reasons from faulty and damaged sensors or energy reduction or DoS attacks.

Physical layer attack. The attackers take over the control of the sensor node by a physical attack. The attackers read the stored data and the ongoing transmission. The attacker can also manipulate the data once it is decoded. This attack can also result in the physical destruction of the node, and the node availability can also be captured on behalf of its data unavailability.

Table 1
Related work on IoT solutions.

Papers	Contribution	Tool/Method
[9]	Protect users' location privacy by clustering the users	SimHash
[8]	Distributed method for clone detection	Watchdog sensors
[10]	Clone detection	Witness-based ERCD protocol
[11]	Clone detection	HIP and HOP
[12]	Sinkhole detection mechanism	Based on RPL routing protocol
[13]	Verifies the legitimacy of the sensor node	RWND
[15]	Clone detection	Witness-based LSCD
[16]	Clone detection	RAND
[18]	Bbotnet-based DDoS attacks detection	Machine learning
[22]	Relay attack detection	Machine learning
[21,24]	Secure the body sensor network, clone detection	Deep learning
[19,19,23,26]	Detect the attacks against IoT, clone detection	Machine learning

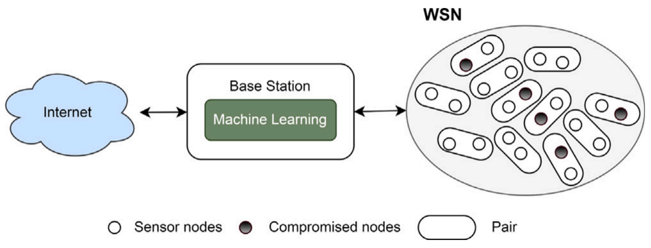

Fig. 2. Proposed solution—Abstract view.

Table 2
Notation list.

n	Total number of sensor nodes in the network
x	Unpaired node selecting the node to be paired
y	Selected node to be paired
ID	Identity information of x
M	Message containing x's private information
Loc	Location of x

3.2. Node pairing approach

The objective of this research is to propose a centralized model to detect the data alteration by making the pair of all sensor nodes in the network, without borrowing any memory or network overhead on the sensor nodes. The queen node is the origin of this security model, thus to detect the attacks, machine learning approaches are applied to the queen node. An abstract view of our proposed solution is shown in Fig. 2, where a WSN has compromised nodes transmitting malicious data. In our sensor pairing approach, we have two following assumptions:

- Each WSN in the IoT network contains an even number of sensors in order to pair all the sensors.
- Each IoT node shares the sensor data it collects by connecting to another edge node or a gateway. Further, this data is processed on a gateway or in the cloud. The machine learning algorithm is applied where the data is processed and analyzed.

Each sensor node collects the event data and forwards it to the queen node. In our model, the queen node; has to accomplish the tasks of attack detection. While collecting the data from the environment, the sensor nodes transmit the observed data to the base station over multi-hop paths. In a witness-based solution, every sensor node needs the extra capacity in buffer storage to become a witness. This storage is used to load the private information of the source sensor node in order to conduct the legitimacy verification. On the other hand, in our solution, the queen node verifies the legitimacy based on the collected event data from sensor nodes in pairs using machine learning approaches. Sensor nodes do not store the private information of any other node so do not need the extra buffer storage. Our proposed solution successfully detects security attacks and does not have memory or network overhead on sensor nodes.

Each sensor node has a unique ID, at the same time a pair ID, where no more than two sensors can have the same pair ID in the network. The queen node keeps a record of the pair ID of each sensor node to compare the collected event data from both sensor nodes and identify the attacks. We distribute all sensor nodes in the network into pairs. The sensors for pairing are selected based on having the shortest path between them. Researchers have proposed many shortest path calculation techniques, such as in [31] the author proposed a fast shortest path distance estimations method. The input of each sensor in pairs will verify the working of the second sensor. In our proposed solution each sensor node works independently, at the same time in pairs in a way that its collected data is used to legitimize its pair sensor node without adding any overhead of communication or memory on it. In other words, each node works as a witness of another node in pairs as its result confirms the working of another node in the pair.

3.2.1. Node pairing algorithm

We propose an algorithm to select the sensor node for pairing. The identity information of the sensor node is encrypted along with its location and a message is formatted to transmit to the neighbor nodes. If the neighbor node (having the shortest path) is vacant and not paired with anyone, then the pair will be made. Else the algorithm increases/decreases by 1 node and checks the next node until it finds the vacant node for pairing. This step is performed for each vacant node to be paired. Algorithm 1 represents the selection process of pairs and Table 2 represents the mathematical symbols used in the algorithm.

Algorithm 1 Algorithm for sensor nodes pair selection

```
1:  $x \leftarrow M(ID_x, loc_x)$
2:  while $x < n$ do
3:    if $(x < x')$ then
4:      $x' \leftarrow ShortestPathCal(x)$
5:      $x++;$
6:    else
7:      $x' \leftarrow ShortestPathCal(x)$
8:      $x--;$
9:    end if
10: end while
11: $y \leftarrow x'$
```

Every sensor has the pair ID, which is transmitted to the queen node along with private information and event information. The queen node analyzes the pair-wise data using the machine learning approaches. It is difficult for an attacker to eavesdrop and identify the pair ID of both sensor nodes in a pair, as it is hard for an attacker to find the structure of the pair within WSN.

Time complexity of algorithm. The time complexity of our algorithm is $O(logx')$, where x' is the selected node to be paired. We have one while loop and the volume of statements executed within each iteration is a constant. So, we just add the number of iterations in order to draw the association between x' and the volume of statements executed.

The body of while loop executes while $x = 2^0, 2^1, 2^2, 2^3, \dots, 2^{\lfloor log_2 n \rfloor}$ and, and this sequence has $1 + \lfloor log_2 n \rfloor = O(logn)$ values.

3.2.2. *Types of node pairing*

There can be two types of pairing: the first is between the same types of sensor nodes (homogeneous pairing) and the second is between different types of sensor nodes (heterogeneous pairing). Further, these two types of pairing can be between sensor nodes based on event-driven input, called passive pairing, or continuous input, called active pairing.

Passive pairing. In this approach, only one sensor node is online and active, while the second sensor node acts passively and remains offline. When an event occurs at a particular time, the first sensor node perceives the environment, transmits data to the base station, and gives an activation call to the second sensor node. The second sensor node in the pair perceives the environment and sends data to the base station. The base station applies machine learning algorithms on both sensor nodes' input to check the accuracy.

Active pairing. Both sensor nodes are online and active in the pair of sensor nodes based on continuous input. Both sensor nodes perceive the environment and transmit data to the base station simultaneously. It can be the homogeneous pairing or heterogeneous pairing. We demonstrated two case studies of heterogeneous sensor node pairing and applied machine learning algorithms to evaluate our solution.

Example. For example, in the automatic defense or remote battlefield systems, face detection cameras, seismic, acoustic, thermal, or magnetic sensors (to observe the ambient energy level) are used for enemy detection and take the appropriate actions. The seismic sensor and acoustic wave sensor can work in pairs to detect ground motion and sound level, respectively, in order to detect the enemy. According to our proposed model, all sensors in the network are divided into pairs. If an attacker compromised a seismic sensor node and the compromised sensor node shows there is no enemy and ultimately wants to deactivate the actuator. The paired acoustic sensor with it will show the current sound level denoting the presence of the enemy. When our queen node will apply the machine learning algorithms, the result will be against the compromised seismic sensor. Thus the queen node will generate an alarm of attack detection and identify the compromised sensor node. The flow chart of active and passive pairing is demonstrated in Figs. 3 and 4, respectively.

3.3. *Machine learning algorithms*

We use three machine learning techniques, CART, SVM, and KNN for security attack detection. These algorithms use the collected data of each pair of nodes to identify the data alteration attack.

Decision tree (CART). The CART algorithm is an optimized version of the decision tree and a powerful machine learning model that we used for both classifications and regression [32]. In our work, CART is used for classification i.e., whether data is malicious or normal. CART built a binary decision tree by splitting a node into two sub-nodes continually, beginning with the parent node. A significant advantage of using CART in our solution is that it considers all possible results of a decision and traces each path to a deduction.

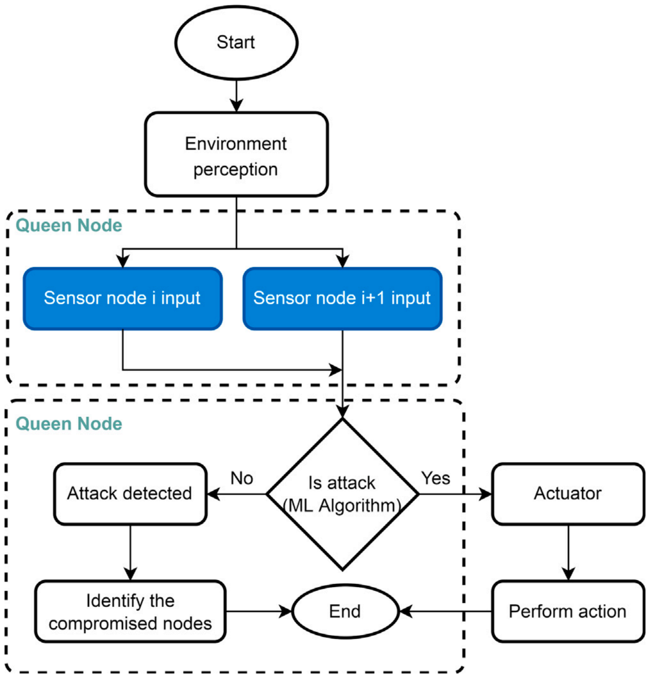

Fig. 3. Proposed solution—Continuous nodes flow chart.

Table 3
SVM kernel functions.

Name	Formula
Radial	$K(x, x_i) = exp(-\alpha\|x, x_i\|^2)$
Polynomial	$K(x, x_i) = (\alpha(x.x_i) + \beta)^d$
Sigmoidal	$K(x, x_i) = tanh(\alpha(x.x_i) + \beta)$
ANOVA	$K(x, x_i) = (\sum_i exp(\alpha(x - x_i))^2)^d$

Support vector machine (SVM). It is a supervised learning model which is widely used for both regression and classification problems. In this research, we have used it for classification and then compare its performance with various algorithms. SVM plot each data record as a point in multi-dimensional space, where dimensions are the number of features we have in our dataset. Then, the classification is performed by finding the hyperplane or a set of hyperplanes to differentiate the classes. SVM attempts to maximize the margin, for example, the distance between the classifier and the nearest training datum.

SVM is a statistical model which performs better in order to avoid local optima than other classification algorithms. SVM is a kernel-based learning algorithm that tries to reach the optimal hyperplane. The most widely used kernel functions in SVM are summarized in Table 3.

Four basic types of SVM kernels are listed in Table 3. We have applied the Radial Basis Function kernel. Radial Basis Function performs efficiently when dealing with nonlinear cases. Classification is tested based on C and $\frac{1}{\sigma^2}$ from the (1), where C > 0 is the penalty of the error term.

$$K(x, y) = exp(-\frac{\|x - y\|^2}{\sigma^2}) \quad (1)$$

K-nearest neighbors (KNN). The nearest neighbor (NN) is one of the most fundamental and effective methods, widely used for classification and regression. M. Pelillo looked back in history and shows that the nearest neighbor rule can be found in Alhazen's influential treatise on optics [33]. The KNN is the structureless model designed by T. M. Cover and P. E. Hart [34]. As the KNN is a lazy algorithm so it does not learn but just stores the training instances during training time. Data is stored in some data structure so that searching becomes faster. The KNN is based on a distance method that calculates the distance of all

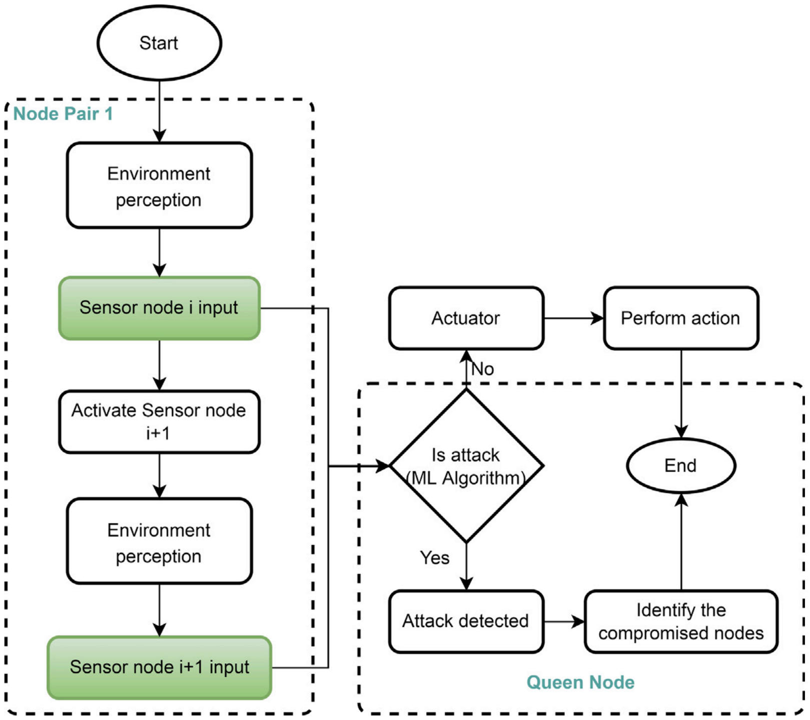

Fig. 4. Proposed solution—Event-driven nodes flow chart.

training data points from sample data and the data point with minimum distance is the nearest neighbor. Various distance functions have been used in KNN to calculate the distance for continuous and discrete variables, such as Euclidian, Chebychev, Manhattan, Minkowski, and Hamming distance measures [35]. The Euclidean distance function is the most extensively used for continuous variables. Let x and y are two points, then the distance (d) from x to y, or from y to x is given in (2).

$$d(x, y) = \sqrt{\sum_{i=1}^{k} (x_i - y_i)^2} \tag{2}$$

The k's value specifies how many nearest neighbors are to be taken into count in order to classify the sample data. If the value of k is 1, then the sample data is just categorized to the class of its nearest neighbor to overcome these issues, the researchers have proposed numerous variances in classical KNN to improve the accuracy and gain the speed and time efficiency [36].

KNN has been used in many applications in statistical pattern recognition, data mining, and many others. In this research, the k nearest neighbors are figured as below:

1. Save all training samples x_i^j in the storage.
2. For the best generalization of the training data, determine the hyperparameter (the optimal value of k). We have a loop through a range of reasonable values (1 to 30) for k and for each value used k fold cross-validations to estimate the out-of-sample accuracy.
3. Measure the distance between query-instance (x) and all training samples x_i^j, given in (3).

$$dis(x, x_i^j)\sqrt{\sum_{i=1}^{d} x(i) - x_i^j(i)^2} \tag{3}$$

4. Find the k-minimum distance between the query-instance (x) and each k $j_{min}^1, j_{min}^2, \ldots, j_{min}^k$
5. Get all categories of training data for the sorted value under k.
6. Find the weighted distance of the query-instance (x) from each of the k nearest points as given in (4).

$$w = 1 - \frac{dis(x, x_i^j)}{\sum_{i=0}^{k} dis(x, x_i^j)} \tag{4}$$

4. Case studies and results

In this section, we provide experiments and results with the help of two case studies in order to evaluate our model. For each case study, we compare the performance of multiple machine learning algorithms by using publically available real-world datasets in terms of attack detection accuracy, recall, precision, training time, and testing time. All machine learning algorithms are applied in Python programming language using the sci-kit learn built-in libraries. In this section, we also provide the memory and energy consumption analysis of our solution.

4.1. Case study 1

Our first case study is from the domain of health care. Patients are more concerned about their protected health information being compromised, but the maximum of the current systems fails to provide solid security to preserve the patient's privacy [37]. The security attack in healthcare may result in severe damage or even the loss of life. To evaluate our solution, we consider the two interlinked sensor nodes i.e., the blood pressure sensor and glucose meter. High glucose level is directly interlinked with elevated blood pressure, which results in an increased risk for hypertension [38]. Security attacks can be deployed to disrupt the functionality of medical actuators like pacemakers, which are small devices used to maintain a suitable heart rate by sending electrical impulses to the heart muscles [39]. In [21], the authors have shown two thread models for glucose monitoring node and insulin pump which results in manipulating and delivering a lethal dose to patients and endanger their lives.

According to our proposed model, the blood pressure sensor and glucose sensor node work in pairs to detect if there is a data manipulation attack. If an attacker compromises the blood pressure sensor and shows the malicious value and wants the patient to take the overdose. The paired glucose sensor with it will show the inverse value of blood pressure which does not demand a high dose. When the queen node will apply the machine learning algorithm on it the result will be against the compromised blood pressure sensor. Thus the queen node will generate an alarm of attack detection and identify the compromised sensor node.

Dataset preparation. To evaluate this case study, we choose the Pima Indian Diabetes dataset from the UCI machine learning repository [40] and applied CART, SVM, and KNN algorithms to it. This dataset is from the National Institute of Diabetes and Digestive and Kidney Diseases. The dataset contains a total of nine features from which one is the target attribute. The purpose of this dataset is to predict whether or not a patient has diabetes, based on the eight features included in the dataset.

Before applying the machine learning algorithm, dataset preparation and cleanness plays a vital role to achieve a good accuracy rate and speed up the learning process. This process includes eliminating the unnecessary dataset features that may reduce the performance and transforming non-numerical features into numerical values and replacing missing values. The data preparation process contains two main steps called, data preprocessing and data normalization.

- **Data Preprocessing**
  Pima Indian diabetes dataset just contains numeric values. We processed the dataset in two different ways. So we will have three types of dataset and accordingly the results, which we called the original dataset, processed dataset 1, and processed dataset 2. The original dataset is the actual download file and applied all three machine algorithms on it in order to compare the performance. In the first type of processing, we retrieve a pair of features and a label. The selected pair features are Plasma glucose concentration level and Diastolic blood pressure (mm Hg). The target attribute (whether or not a patient has diabetes) is calculated based on these two features.
  In the second type of processing, the same pair of features are selected which are Plasma glucose concentration level and Diastolic blood pressure. The target attribute is calculated based on these two features such as if glucose level is or blood pressure is greater than the normal value then the patient needs the dose. So the target attributes' value will be 1 otherwise no need for a dose and the value is 0. The normal values of glucose and blood pressure are 155 and 80, respectively. Moreover, 5% of target variable data is manipulated. So both of our preprocessed datasets contain just three features and the dataset file size is reduced from 107 kb to 31 kb (3.4 times less in memory).
- **Data Normalization**
  Some attributes in the dataset have larger values as compared to others. This can lead to incorrect results since a machine learning algorithm might be biased to the larger feature values. We normalize the dataset by converting the outweighing attributes with small values. We scale the attributes within the range between [0.0, 1.0]. A min–max normalization method is used to scale all dataset attributes values within [0.0, 1.0], as shown in the below equation:

  $$b = \frac{a - a_{min}}{a_{max} - a_{min}} \tag{5}$$

  where a is the original value, b is the normalized value, and a_{max} and a_{min} are the minimum and maximum values of the attributes, respectively.

Results. The dataset contains 768 records which are provided by the National Institute of Diabetes and Digestive and Kidney Diseases. 80% of the dataset records were used for training and the rest of the 20% for testing. Machine learning algorithms are applied to the original Pima Indian Diabetes dataset (9 attributes) and both processed datasets which contain the data of selected pair features (3 attributes). The CART, SVM, and KNN algorithms are applied to all three datasets. Accuracy, recall, precision, f-score, training time, and prediction time are used to evaluate the effectiveness of machine learning models. These metrics are calculated as below:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

$$F - Score = 2 * \frac{(Recall * Precision)}{Recall + Precision} \tag{9}$$

The accuracy illustrates the overall performance of the models as the classification of all benign and attack observations. The highest accuracy is achieved by CART which is 100% on processed dataset 2. While SVM and KNN achieved 80.8% and 96% accuracy rates on processed dataset 2, respectively. The train and test time on all algorithms on processed datasets 1 and 2 is faster as compared to the original dataset time. It is nearly twice faster. Table 4 summarizes the results of the CART, SVM, and KNN algorithms on the original diabetes dataset and processed datasets.

Table 4
Machine learning models results on Pima Indian diabetes dataset.

Models	Dataset	Accuracy	Recall	Precision	F-score	Train time	Test time
CART	Original	72.5	57.4	62	59.6	0.004	0.001
	Processed 1	70.5	50	60	54.5	<0.001	0.010
	Processed 2	100	100	100	100	0.026	<0.001
SVM	Original	73.8	38.8	75	51.2	0.015	0.003
	Processed 1	73.8	40.7	73.3	52.3	0.020	<0.001
	Processed 2	80.8	44	100	61.5	0.035	0.005
KNN	Original	75.8	55.5	69.7	61.8	0.001	0.004
	Processed 1	73.2	48.1	66.6	55.9	<0.001	<0.001
	Processed 2	96	92.5	96.1	94.3	0.034	0.002

4.2. Case study 2

Our second case study is from the domain of environmental monitoring-related events. Agriculture farmers need an automatic system to water their crops. IoT has the answer: an automatic plant irrigation system accommodates agriculturalists for irrigating the crops. The sensor nodes sense the rain and temperature and automatically turn on/off the water pump when it is required. According to our proposed model, the rain and temperature data are used in pairs to detect if there is an attack.

If one of any sensor nodes is under attack, behave maliciously and transmit altered data to the queen node. Our paring approach detects such types of irregularities. In [41], the author highlighted the security issues in environmental monitoring sensors, such as the temperature, rain, and light sensors. For example, an attacker compromised a temperature sensor and showed the high-temperature value, and wanted the water pump to remain turned on. When the queen node will apply the machine learning algorithm on it the result will be against the compromised temperature sensor. Thus the queen node will generate an alarm of attack detection and identify the compromised sensor node.

Dataset preparation. To evaluate this case study, we have used the real weather dataset of Australia provided by the Bureau of Meteorology, Australian Government [42] and applied CART, SVM, and KNN algorithms to it. This dataset is also available at Kaggle [43]. The dataset contains the record of Australian weather data from 2007-10-31 to 2017-06-24. The dataset holds a total of 23 features from which one is the target variable. RainTomorrow is the target variable to predict, it means—did it rain the next day, Yes or No? The data preparation includes data preprocessing and data normalization.

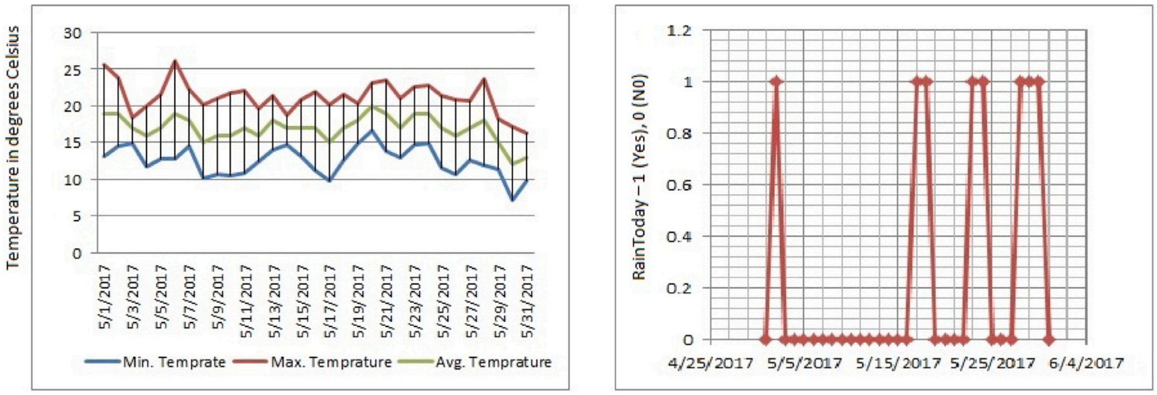

Fig. 5. One month sample of Sydney temp. (left) and rain (right) - Rain of Australia dataset.

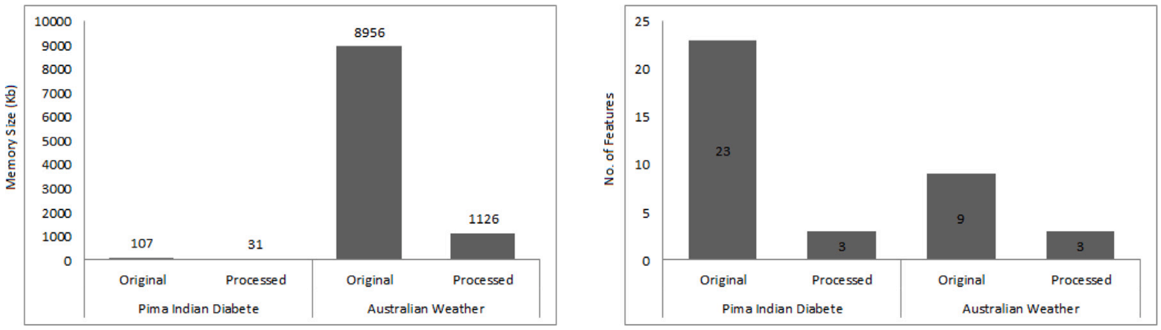

Fig. 6. Memory and attributes analysis of datasets.

- **Data Preprocessing**
The dataset contains unnecessary dataset features, non-numeric values, and missing values. The dataset is processed to eliminate the unnecessary features that may reduce the performance and transforms non-numerical features into numerical values and replace the missing values with zero. We eliminate five columns that may reduce the performance, which is Date, Location, WindGustDir, WindDir9am, and WindDir3pm. We converted the non-numerical values in the whole dataset with numerical values. We replace NA, Yes, and No with 0, 1, and 0, respectively.
The dataset is also processed in two ways. Similarly, we will have three types of datasets and accordingly the results, which we called the original dataset, processed dataset 1, and processed dataset 2. In the first type of processing, we retrieve a pair of features and a label. The selected pair features are Evaporation and RainToday. The target attribute RainTomorrow is calculated based on these two features.
In the second type of processing, the selected features for pairing are average temperature and RainToday attributes. We have calculated the average temperature based on minimum temperature and maximum temperature. The target attribute is calculated based on these two features such as if the average temperature is greater than or equal to 15 and It is not raining today (Rain = 0), then the target attribute's value will be 1 otherwise no need for dose and the value is 0. Moreover, 5% of target variable data is manipulated. Fig. 5 shows the one-month sample of Sydney temperature and rain data. So both processed datasets contain just three features and the dataset file size is reduced from 8956 kb to 1126 kb (8 times less in memory), as depicted in Fig. 6. We focused on the five top populated cities of Australia and then evaluated our model on the whole Australian weather dataset. The detail of dataset records is shown in Table 5.
- **Data Normalization**

Table 5
Weather of Australia dataset detail.

	Total records	Train Rec. (80%)	Test Rec. (20%)
Sydney	3327	2661	666
Melbourne	2298	1838	460
Brisbane	3126	2500	626
Perth	3192	2554	638
Adelaide	3019	2415	604
Australia	140 086	112 068	28 018

We normalize the dataset by converting the outweighing attributes with small values. We scale the attributes within the range between [0.0, 1.0]. A min–max normalization method is used to scale all dataset attributes values within [0.0, 1.0], as shown in (5).

Results. The dataset contains 145 460 records which are provided by the Bureau of Meteorology, Australian Government. 80% of the dataset (116 368 records) was used for training and the rest of the 20% (29 029 records) for testing. Machine learning algorithms are applied to the original dataset (23 attributes) and the 2 processed datasets (3 attributes). The CART and KNN algorithms are applied to all three datasets. The large train time of SVM makes it inappropriate for this dataset. Same performance evaluation measures are applied to this dataset as applied to Pima Indian diabetes dataset.

The highest accuracy is achieved by CART which is 100% on processed dataset 2. While KNN achieved 99.9% accuracy rates on processed dataset 2. The train and test time on all algorithms on processed datasets 1 and 2 is faster as compared to the original dataset time. The results of CART and KNN algorithms on the original and processed dataset 1 are summarized in Table 6. While the city-wise processed dataset results of Australian weather are summarized in Table 7.

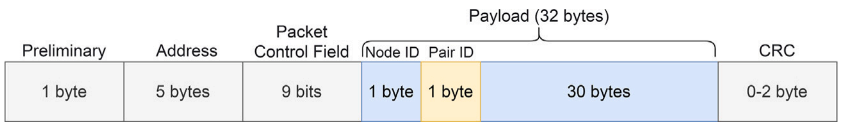

Fig. 7. Data Packet Frame of WSN Node.

Table 6
Machine learning models results on Australian weather dataset (Original and processed dataset 1).

Models		Accuracy	Recall	Precision	F-score	Train time	Test time
CART	Original	78.8	51.8	47	49.3	2.042	0.010
	Processed 1	80.19	2	56	3	0.069	0.003
KNN	Original	85	48	67.2	56	2.542	52.528
	Processed 1	78.8	30.8	45.2	36.7	7.839	10.863

Table 7
Machine learning models results on city wise distributed Australian weather dataset (Processed dataset 2).

Location	Model	Accuracy	Recall	Precision	F-score	Train time
Sydney	CART	100	100	100	100	0.033
	KNN	99.6	1	99.5	99.7	0.082
Melbourne	CART	100	100	100	100	0.030
	KNN	100	100	100	100	0.061
Brisbane	CART	100	100	100	100	0.031
	KNN	99.3	100	99.2	99.5	0.078
Perth	CART	100	100	100	100	0.031
	KNN	99.6	100	99.5	99.7	0.076
Adelaide	CART	100	100	100	100	0.032
	KNN	98.6	100	97.9	98.9	0.073
Australia	CART	100	100	100	100	0.485
	KNN	99.9	100	99.9	99.9	27.1

4.3. Memory and energy consumption

This section illustrates the memory and energy consumption analysis of our solution. Three types of cost are measured for our proposed model as follows:

Memory cost. Memory cost in a witness base solution is measured based on the number of witnesses' location claims that are stored by each sensor node in the WSN. In our proposed model, the sensor node contains only its own pair ID. The queen node contains the pair IDs of all sensor nodes. The memory cost (C_m) of the queen node can be calculated by multiplying the total number of sensor nodes N_n by the size of location (L). It can be represented as below:

$$C_m = N_n.L \tag{10}$$

Communication cost. The communication cost is the number of witnesses claimed that are transmitted and received by each sensor node during detection. In our proposed model pair ID is embedded in the data frame and transmitted along with the node ID and information. The structure of the sensor node data packet based on the nRF24L01 wireless protocol is illustrated in [44]. The payload consists of 32 bytes from which nine bytes are used. The first byte contains node ID. Sensor data is stored in the next seven bytes. The 9th byte contains battery status. We embed 1 byte for pair ID in the data frame, as depicted in Fig. 7. The communication costs can be of two types:

- *Communication Cost—Sensor Node to The Queen Node:* It is based on the forwarding of the location of a sensor node from the sensor node to the queen node in the network. Communication cost (C_{sn}) can be calculated based on the size of location (L), transmission cost of bits (T_b), and the average path length between the sensor node and the queen node (Len) and can be stated as follows:

$$C_{sn} = L.T_b.Len \tag{11}$$

- *Communication Cost—Selecting Witness Sensor Nodes:* The witness selection cost is based on the random walk called C_w. It is based on the size of the location (L), the number of random walks (N_r), the number of walk steps (N_t), and the transmission cost of sending bits (T_b). It is represented as below:

$$C_w = L.N_r.N_t.T_b \tag{12}$$

Total communication cost (C_{tot}) can be obtained as below:

$$C_{tot} = C_{sn} + C_w \tag{13}$$

Computational cost. During the detection process, not a single computation is performed on the sensor node. Computational cost is calculated based on the number of attack detection verification performed by the queen node during the detection process. The results section of both case studies shows the training speed of machine learning algorithms to detect attacks.

5. Conclusion

We proposed a novel pairing structure to confirm the legitimacy of sensor nodes' data along with a pairing algorithm. We provide two case studies to evaluate our proposed solution and accordingly perform experiments on two publically available real-world datasets. The CART, SVM, and KNN algorithms are applied to both datasets in terms of attack detection accuracy, recall, precision, f-score, training time, and prediction time. We processed the datasets in two different ways. Both of our processed datasets contain lesser attributes (just three) and the dataset file size is reduced up to 8 times lesser in memory than the original dataset memory size. The highest accuracy is achieved by CART which is 100% on processed dataset 2. We provide the memory and energy consumption analysis to show that the proposed method has lower communication and memory overhead on IoT nodes. In future research, we would like to use sensor pairing techniques combined with deep learning approaches to enhance performance.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] P. Sharma, S. Jain, S. Gupta, V. Chamola, Role of machine learning and deep learning in securing 5G-driven industrial IoT applications, Ad Hoc Netw. 123 (2021) 102685.
[2] I. Butun, P. Österberg, H. Song, Security of the Internet of Things: Vulnerabilities, attacks, and countermeasures, IEEE Commun. Surv. Tutor. 22 (1) (2019) 616–644.
[3] M.M. Salim, S. Rathore, J.H. Park, Distributed denial of service attacks and its defenses in IoT: a survey, J. Supercomput. 76 (7) (2020) 5320–5363.
[4] L. Cui, S. Yang, F. Chen, Z. Ming, N. Lu, J. Qin, A survey on application of machine learning for Internet of Things, Int. J. Mach. Learn. Cybern. 9 (8) (2018) 1399–1417.
[5] B.K. Mohanta, D. Jena, Internet of things security using machine learning, in: Advances in Machine Learning and Computational Intelligence, Springer, 2021, pp. 129–136.

[6] Y. Li, Y. Zuo, H. Song, Z. Lv, Deep learning in security of internet of things, IEEE Internet Things J. (2021).
[7] R. Taylor, D. Baron, D. Schmidt, The world in 2025-predictions for the next ten years, in: 2015 10th International Microsystems, Packaging, Assembly and Circuits Technology Conference (IMPACT), IEEE, 2015, pp. 192–195.
[8] S.S. Bhunia, M. Gurusamy, Dynamic attack detection and mitigation in IoT using SDN, in: 2017 27th International Telecommunication Networks and Applications Conference (ITNAC), IEEE, 2017, pp. 1–6.
[9] Q. Zhang, Y. Zhang, C. Li, C. Yan, Y. Duan, H. Wang, Sport location-based user clustering with privacy-preservation in wireless IoT-driven healthcare, IEEE Access 9 (2021) 12906–12913.
[10] Z. Zheng, A. Liu, L. Cai, Z. Chen, X. Shen, Energy and memory efficient clone detection in wireless sensor networks, IEEE Trans. Mob. Comput. 15 (5) (2015) 1130–1143.
[11] M. Conti, R. Di Pietro, A. Spognardi, Clone wars: Distributed detection of clone attacks in mobile WSNs, J. Comput. System Sci. 80 (3) (2014) 654–669.
[12] M. Zaminkar, R. Fotohi, SoS-RPL: securing internet of things against sinkhole attack using RPL protocol-based node rating and ranking mechanism, 2020, arXiv preprint arXiv:2005.09140.
[13] W.Z. Khan, M.Y. Aalsalem, N. Saad, Distributed clone detection in static wireless sensor networks: random walk with network division, PLoS One 10 (5) (2015) e0123069.
[14] J. Huang, X. Li, C.-C. Xing, W. Wang, K. Hua, S. Guo, DTD: A novel double-track approach to clone detection for RFID-enabled supply chains, IEEE Trans. Emerg. Top. Comput. 5 (1) (2015) 134–140.
[15] M. Dong, K. Ota, L.T. Yang, A. Liu, M. Guo, LSCD: A low-storage clone detection protocol for cyber-physical systems, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 35 (5) (2016) 712–723.
[16] W.Z. Khan, M.S. Hossain, M.Y. Aalsalem, N.M. Saad, M. Atiquzzaman, A cost analysis framework for claimer reporter witness based clone detection schemes in WSNs, J. Netw. Comput. Appl. 63 (2016) 68–85.
[17] T. Saha, N. Aaraj, N. Ajjarapu, N.K. Jha, SHARKS: Smart hacking approaches for risk scanning in internet-of-things and cyber-physical systems based on machine learning, IEEE Trans. Emerg. Top. Comput. (2021).
[18] S. Pokhrel, R. Abbas, B. Aryal, IoT security: Botnet detection in IoT using machine learning, 2021, arXiv preprint arXiv:2104.02231.
[19] B. Parno, A. Perrig, V. Gligor, Distributed detection of node replication attacks in sensor networks, in: 2005 IEEE Symposium on Security and Privacy (S&P'05), IEEE, 2005, pp. 49–63.
[20] A. Demontis, M. Melis, B. Biggio, D. Maiorca, D. Arp, K. Rieck, I. Corona, G. Giacinto, F. Roli, Yes, machine learning can be more secure! a case study on android malware detection, IEEE Trans. Dependable Secure Comput. 16 (4) (2017) 711–724.
[21] U. Ahmad, H. Song, A. Bilal, S. Saleem, A. Ullah, Securing insulin pump system using deep learning and gesture recognition, in: 2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE), IEEE, 2018, pp. 1716–1719.
[22] U. Ahmad, H. Song, A. Bilal, M. Alazab, A. Jolfaei, Securing smart vehicles from relay attacks using machine learning, J. Supercomput. 76 (4) (2020) 2665–2682.
[23] A. Sheneamer, J. Kalita, Semantic clone detection using machine learning, in: 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA), IEEE, 2016, pp. 1024–1028.
[24] J. Ghofrani, M. Mohseni, A. Bozorgmehr, A conceptual framework for clone detection using machine learning, in: 2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI), IEEE, 2017, pp. 0810–0817.
[25] H. Xue, G. Venkataramani, T. Lan, Clone-hunter: accelerated bound checks elimination via binary code clone detection, in: Proceedings of the 2nd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages, 2018, pp. 11–19.
[26] J. Svajlenko, C.K. Roy, A machine learning based approach for evaluating clone detection tools for a generalized and accurate precision, Int. J. Softw. Eng. Knowl. Eng. 26 (09n10) (2016) 1399–1429.
[27] M.M. Rana, IoT-based electric vehicle state estimation and control algorithms under cyber attacks, IEEE Internet Things J. 7 (2) (2019) 874–881.
[28] M. Etemadi, M. Ghobaei-Arani, A. Shahidinejad, Resource provisioning for IoT services in the fog computing environment: An autonomic approach, Comput. Commun. 161 (2020) 109–131.
[29] M.S. Aslanpour, S.E. Dashti, M. Ghobaei-Arani, A.A. Rahmanian, Resource provisioning for cloud applications: a 3-D, provident and flexible approach, J. Supercomput. 74 (12) (2018) 6470–6501.
[30] M. Rana, Architecture of the internet of energy network: An application to smart grid communications, IEEE Access 5 (2017) 4704–4710.
[31] M. Potamias, F. Bonchi, C. Castillo, A. Gionis, Fast shortest path distance estimation in large networks, in: Proceedings of the 18th ACM Conference on Information and Knowledge Management, 2009, pp. 867–876.
[32] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140.
[33] M. Pelillo, Alhazen and the nearest neighbor rule, Pattern Recognit. Lett. 38 (2014) 34–37.
[34] T.M. Cover, P.E. Hart, Nearest neighbor pattern classification, IEEE Trans. Inform. Theory 13 (1967) 21–27.
[35] M.R. Abbasifard, B. Ghahremani, H. Naderi, Article: A survey on nearest neighbor search methods, Int. J. Comput. Appl. 95 (25) (2014) 39–52, Full text available.
[36] N. Bhatia, Vandana: Survey of nearest neighbor techniques, Int. J. Comput. Sci. Inf. Secur. 8 (2) (2010) 302–305.
[37] P. Gope, T. Hwang, BSN-Care: A secure IoT-based modern healthcare system using body sensor network, IEEE Sens. J. 16 (5) (2015) 1368–1376.
[38] S. Reule, P.E. Drawz, Heart rate and blood pressure: any possible implications for management of hypertension? Curr. Hypertens. Rep. 14 (6) (2012) 478–484.
[39] D. Clery, Could your pacemaker be hackable?, 2015.
[40] Pima Indian Diabetes Database, Url: www.ics.uci.edu/~mlearn/MLRepository.html.
[41] S.A. Kumar, T. Vealey, H. Srivastava, Security in internet of things: Challenges, solutions and future directions, in: 2016 49th Hawaii International Conference on System Sciences (HICSS), IEEE, 2016, pp. 5772–5781.
[42] Bureau of Meteorology, Australian Governemnt, Url: http://www.bom.gov.au/.
[43] Rain in Australia, Url: https://www.kaggle.com/jsphyg/weather-dataset-rattle-package.
[44] S.K. Gharghan, R. Nordin, M. Ismail, Energy efficiency of ultra-low-power bicycle wireless sensor networks based on a combination of power reduction techniques, J. Sensors 2016 (2016).

Usman Ahmad is a final year Ph.D. student in the School of Computer Science and Technology at Beijing Institute of Technology. His research interests center around the role of Machine Learning and Artificial Neural Networks in Cybersecurity and IoT. He is also concerned about the architecture and methods used in Deep Learning models. He holds a Master's degree in Computer Science and has several publications.