Journal of Reliable Intelligent Environments (2024) 10:73–91
https://doi.org/10.1007/s40860-023-00202-y

ORIGINAL ARTICLE

Formal verification for security and attacks in IoT physical layer

Zinah Hussein Toman[1,2] · Lazhar Hamel[3] · Sarah Hussein Toman[1,2] · Mohamed Graiet[4] · Dalton Cézane Gomes Valadares[5]

Received: 23 June 2022 / Accepted: 22 March 2023 / Published online: 6 May 2023

Abstract
IoT devices are more important than ever. In a connected world, IoT devices have many uses. They are no longer merely used at work; they are part of our everyday lives. Security concerns arise if the devices generate, collect, or process sensitive data. Physical layer security controls are the cornerstone once the risk for humans increases when physical security fails. To achieve security in IoT devices, preventing is better than detecting. Formal verification is an important and valuable tool for detecting possible vulnerabilities and ensuring data security. Thus, this paper proposes an Event-B proof-based formal model of IoT physical layer security and attacks from the requirements analysis level to the goal level. Our model is built incrementally using a refining method during design and verification. We present a three-level formal approach: first, the construction of the IoT physical layer; then, we check for IoT physical layer vulnerabilities by processing the lack of some characteristics that cause these vulnerabilities, such as speed, typical bandwidth, and power consumption; lastly, we detect physical layer attacks like jamming and MAC spoofing, which helps to build security proofs. Our approach uses an electrocardiogram (ECG) IoT system as a case study, and as an additional case study to back up the proposed method's generalizability, we used a fire alarm system. Also, we use the proof obligations and the ProB animator in the Rodin model checking tool to check and validate our approach.

Keywords Internet of Things (IoT) · Event-B · Formal model · Physical layer security · Attacks

1 Introduction

The Internet of Things (IoT) is a term that refers to physical things connected to the Internet and their virtual representation. It includes not just human participation but also "Things". The IoT promises to bring new services to people in all aspects of our society. It intends to deliver pervasive connection and data collection capabilities across the home, vehicle, and industrial environments. Hundreds of billions of physical objects are equipped with various sensors and actuators and connected to the Internet through heterogeneous communication systems [1]. In a typical IoT scenario, sensors collect data using smart devices, process these data using controllers to make decisions, and then transmit this information to devices or people for execution [2].

The difficult challenge in this context is verifying and ensuring the functionality, security, and confidentiality of the IoT despite the present and hidden vulnerabilities of connected things and the expanded ineffectiveness of cybersecurity.

In the technology field, IoT security has become an essential topic of conversation [3]. Security weaknesses, technical limits, and software vulnerabilities are among the significant challenges for this technology to take hold in essential applications. Whether innocent or malicious, human error is the most common cause of security vulnerabilities [4]. Moreover, detecting online cyberattacks on IoT devices is challenging due to their limited battery life and computational power. Therefore, we need an alternative method of reducing the attack surface to decrease the threat of an attack.

✉ Zinah Hussein Toman
zinah.hussein@qu.edu.iq

1 Department of Computer Science, FSM, Al-Monastir University, Monastir, Tunisia

2 Department of Computer Science, Faculty of Computer Science and Information Technology, Al-Qadisiyah University, Al Diwaniyah, Iraq

3 Department of Computer Science, ISIMM, Al-Monastir University, Monastir, Tunisia

4 IMT Atlantique, LS2N, Nantes, France

5 Federal Institute of Pernambuco/Federal University of Campina Grande, Campina Grande, Brazil

This method would necessitate more powerful security testing on the device prior to deployment. For this reason, in this paper, we focused on the attacks and security of the IoT's physical layer using formal methods.

In physical layer security for IoT systems, the secrecy rate is the speed at which information can be sent secretly from the source (an IoT device) to its intended destination through communication channels [5]. Using its properties, the physical layer may also provide information-theoretic security, even if the attacker has infinite processing power [6]. IoT devices and communication channels are easy targets for attackers because of their spread-out nature. Many of these devices and protocols also have limited resources, which makes them even more vulnerable. Because they do not get updated often, many of these devices can be hacked or attacked in other ways [7].

The attacker must have physical access to the device or cause a fault in it to stop it from working and get secret information from its side channels, such as how much power and energy it uses, how much radiation it puts out, and how long it takes to run. An attacker may cause a competing action to be performed on the same item or feature of the environment. This is called a cascade attack [8]. Attackers could use these problems and vulnerabilities to gain physical access to IoT devices.

In this paper, we focused on the attacks on the IoT physical layer, such as jamming and MAC spoofing [9]. Jamming is a type of denial of service (DoS) attack in which hostile nodes interfere on the networks to stop legal communication from happening. For instance, an enemy can use the same frequency as the IoT device to stop the communication between IoT devices and gateways. Spoofing is another sort of assault that sends out a false signal. A MAC spoofing attack can make a node appear legitimate. MAC spoofing is the modification of a network interface controller (NIC) card's MAC address.

Preventing security breaches in IoT devices is an alternative to detecting them [10]. This prevention necessitates a reduction in the number of possible attack paths. Promising design approaches, secure coding, static analysis, and comprehensive testing have traditionally been used to accomplish this. However, the security of these methods cannot be guaranteed, and hence, more robust techniques are required. The formal verification application to ensure security is a promising direction, making sure that a design is correct using some mathematical and logical methods.

In the IoT, formal methods are used to do many tasks that directly relate to our everyday lives. They also validate security guarantees and real-time attributes for a given model. Through formal methods, mathematics and logic can be used to check and specify the accuracy of designs. The correct code can be created or implemented from the design once a suitable design has been established [11].

So, this paper proposes a new formal model of IoT physical layer security and attacks based on the Event-B proof-based formal model. Event-B offers rigorous mathematical reasoning that aids in developing software with greater confidence. This model goes from the abstract to the target level incrementally using the refinement method. We outline the paper's major contributions below:

- We propose a formal approach to representing the physical layer of an IoT system.
- We verify the lack of IoT physical layer vulnerabilities as they are essential for formal verification due to their crucial use in conjunction with security-sensitive features of the device. As a result, complete testing of devices and attack detection becomes realistically difficult.
- We use formal verification to detect physical-layer attacks like jamming and MAC spoofing. Formal verification is of great assistance in establishing security proofs.

To verify that the IoT physical layer model is both functional and accurate, we incorporated an electrocardiogram (ECG) IoT system into our model and we tested the model's events to prove there is no problems at any level. The fire alarm system is used as another case study to prove that the proposed method can be applied to a wide variety of situations. We made use of the Rodin model checker and proof obligations. This paper's remaining sections are structured as follows: Sect. 2 introduces related works; Sect. 3 describes the concept of IoT, Physical layer security and attacks and gives an outline of Event-B theories; Sect. 4 Offers apropsed model and motivating examples; Sect. 5 presents the formalization of IoT physical layer in the Event-B model; Sect. 6 proposes a verification and validation of our approach and Sect. 7 concludes this paper.

2 Related work

For a number of years, a considerable amount of work has been spent on studying IoT technology. However, the majority of contributions focus on using formal methods in IoT physical layer security. This subsection provides a chronologically ordered literature evaluation of some of these publications. In this study [12], the authors suggested a safety protocol to address the security weaknesses in medical IoT communication. The suggested software-based communication protocol utilized inter-device cross-authentication and encryption to prevent a variety of attacks, and it was validated with Casper/FDR, a formal verification tool for processes. The results of the test showed that the proposed protocol makes sure that wireless connections between medical devices are safe. In [13], the authors came up with a new

Table 1 The comparison of existing research studies with our approach

References	year	IoT topic	Formal methods	method/tool
[12]	2019	Security and attacks	Model checking	Casper/FDR
[13]	2016	Security	Model checking	SPIN tool and PROMELA language
[14, 15]	2017	Security and attacks	theorem prover	Isabelle
[16]	2017	Security	Process algebra	AVISPA proves
[17]	2017	Security	Model checking	SMV language and the NuSMV model checker
[18]	2017	Security	Model checking	Alloy Analyzer
[19]	2017	Security	Model checking	PRISM
[21]	2017	Security	Model checking	Z3 SMT solver
Our approach		Security and attacks	Model checking	Event-B

way to combine the security parts of IoT devices while taking into account their non-functional properties. A technique and a software tool were used to check the information flow security policy. Attention was paid to certain types of differences between the rules, sensors were tested for strange data, and a few types of security component conflicts were found. The SPIN tool and the PROMELA language were used to check the data flows in a network, verify the security policy, and prove that the method was sound.

It was introduced in [14, 15] new formal modeling by integrating previous formal methods to model actors, devices, and policies of human-centric infrastructures for IoT healthcare systems to investigate security and safety threats. The authors utilized Isabelle, an interactive theorem prover that provides modeling and evaluation of human-centric infrastructures by means of attack path analysis.

The primary contribution made by the authors of this research [16] was the proposed multi-factor remote user verification protocol. Passwords, smart devices, and biometrics are all used in this scheme to ensure the safety of user identities. The authors then presented a formal and informal security analysis of the protocol. Finally, other protocols' computational and communication abilities were compared. The research in [17], presented a method that employs time-aware computations to allow the controller to assess the quality of the entered data. In industrial automation applications, the Cyber-Physical Agnosticism (CPA) property of the Internet of Things (IoT) is verified through formal verification. To simplify the structure of the SMV model, an abstract model of the facility was developed, consisting of four instances of state machines, one elevator car, and three doors. The model of an abstract plant is turned into the SMV language and the NuSMV model checker.

The Secure Swarm Toolkit (SST) was introduced in this paper [18], the authors create to facilitate the development of an IoT authorization service infrastructure. As a result of SST's flexible security options, we anticipate that a wide variety of IoT devices, from sensor nodes to electric power grid control systems, will be able to be integrated into the authorization infrastructure. This research used the automated verification tool Alloy Analyzer to conduct a formal analysis of the system's security.

To automate verification and probabilistically quantify attack probabilities against standard IoT system configurations, the study in [19], introduced a novel IoT Risk Analyzer framework. One can accurately assess the root cause and severity of security risks with the help of a formal model-driven verification approach, which checks all possible behaviors of the reference model in a finite state space. The proposed framework generates MDP models, which are then used by the PRISM model checker to perform automated analysis of system-level risk profiles.

In this work [20, 21], centered on utilizing the Shibboleth protocol for secure data access and outsourcing in a cloud-IoT network. This research proposes the Shibboleth-based Fog-IoT network, which includes Shibboleth to ensure security between Fog Client and Fog Node. The authors formally validated Shibboleth using High Level Petri Nets (HLPN), a mathematical and graphical modeling tool, to demonstrate the correctness of Shibboleth with respect to specific security properties. In addition, the Z3 SMT solver was used to analyze the rules of data flows, proving the system's correctness versus the three security properties. The purpose of this paper is to present a new method that uses the Event-B modeling approach to overcome these verification limitations by modeling systems with abstraction. In comparison to earlier works, while there is a dearth of research to verify the IoT systems that deal with the physical architecture of IoT layers with security and attacks.

Our proposed model integrates modeling and verification of physical layer security and attacks in the IoT, to the advantage of working in both fields, thus contributing to their development. In this paper, we present new work in four ways: we propose a formal verification of the IoT physical layer; we check that the IoT physical layer does not have any security holes; we find physical-layer attacks, and our work is based on a formal model. Table 1 compares existing studies

to our methodology based on a similar topic, methodology, and domain.

3 Background

3.1 IoT concept

The term "Internet of Things" (IoT) is currently a hot topic. Kevin Ashton, who worked for Procter and Gamble, came up with the term in 1999 to describe how he put RFID tags and sensors on items in the supply chain [22]. Since then, RFID has gotten a lot of attention in academia and business. IoT is primarily concerned with the idea of connecting things to a network without human involvement, which facilitates the exchange and gathering of data throughout the network. This is done by connecting gadgets and "things" to the Internet and other networks in different ways. Many things are connected to each other through their built-in actuators and sensors, which can sense their surroundings and gather data, which is then sent to the next IoT layer. This section provides a basic overview of the various IoT layers [23, 24].

1. **Physical layer:** Sensors, actuators, and devices make up this layer. Sensing systems detect IoT environment parameters. It turns sensing technologies' physical data into digital signals for communication and transmission. When some objects are not perceived, embedded sensors can help. So, embedded sensors help the perception layer by processing data at end devices.
2. **Transport layer:** Sensor data from the perception layer can be transferred to the middleware layer via wired or wireless communication routes, and the reverse is also true.
3. **Middleware layer:** Functions in a critical way. An enormous amount of data is received by this layer from the transport layer and processed by this layer. This layer can also be used to manage and control access to the devices and to find new ones.
4. **Application layer:** Application service delivery, which originates from the middleware layer and is delivered to various applications, is the responsibility of this layer. Smart cities, smart homes, smart healthcare, etc. are only a few examples.

As a basis for our work, we retrieved the physical layer in this paper (see Fig. 1). So, to construct an efficient and dependable IoT physical layer formal model, we must first extract the appropriate architectural criteria for this layer. The Table 2 illustrates the functional needs that will be utilized in our formal model:

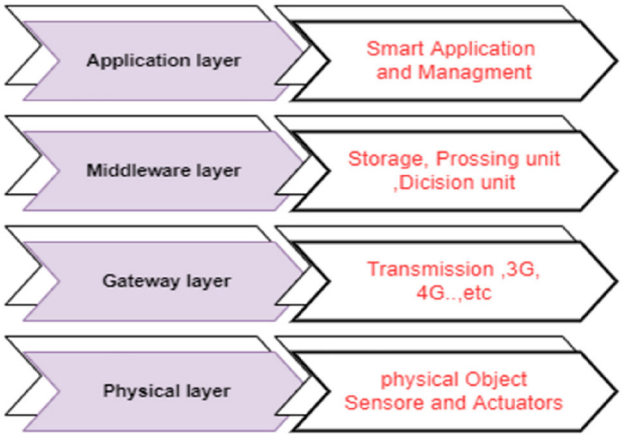

Fig. 1 The IoT system layers architucture

Table 2 The physical layer requirements

Phy-Req1	There is a name, a type, a location, and properties associated with each thing
Phy-Req2	Sensors, actuators, and devices connected to the IoT system are the physical layer components
Phy-Req3	All of the sensors produce data
Phy-Req4	We are able to add data from any IoT device in the physical layer of our system
Phy-Req5	Within the physical layer, we have the ability to add a single or several sensors
Phy-Req6	Within the physical layer, we have the ability to add a single or several actuators
Phy-Req7	The physical layer is responsible for collecting data from various sensors and devices

3.2 Physical layer security and attacks

3.2.1 Physical layer security

The physical layer is the foundation of all security controls. Other security measures can fail without causing a disaster, but when physical security fails, people are usually left completely open to danger [4]. Physical layer security in the IoT has only recently emerged as a solution for enhancing IoT system security. For the first time, privacy is achieved using the physical layer of the network system instead of traditional cryptographic methods like interference and channel security, as well as thermal noise and other things.

Physical layer security in the IoT has only recently emerged as a solution for enhancing IoT system security. Since many sensors and actuators are resource-restricted, it is difficult to incorporate security in IoT systems. This kind of security does not rely on encryption but on the physical aspects of wireless channels, such as fading, noise, interference, and so on [4]. Security is guaranteed by Physical layer security, regardless of the eavesdropper's computational power. So, it is becoming a good alternative or addition

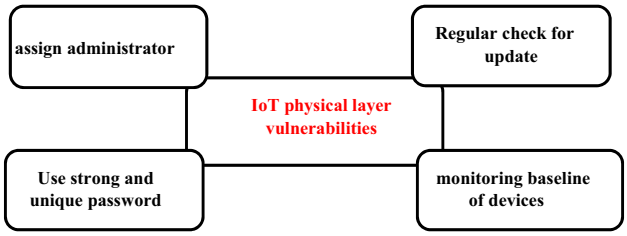

Fig. 2 The proposed characteristics that cause the IoT physical layer vulnerabilities in our model

to hard-to-understand cryptographic algorithms for secure communications in the IoT [25].

In the physical layer, vulnerabilities are directly related to physical access to networks and devices, such as unauthorized network access, data damage, and keystroke logging. An attacker may use one or more of the following physical layer security vulnerabilities to compromise IoT devices [26]:

1. There may be a flaw in a device's functionality. For example, an operation that was supposed to be there but was either left out or only partly done.
2. Even if the necessary security actions have been implemented in the device, implementation defects may exist. Such a defect can be an issue with cryptographic algorithms in particular.
3. Programming bugs like format-string vulnerabilities, buffer overflows, and arithmetic overflows and underflows can be used to make malicious payloads that stop execution and break security.
4. Attackers can trick other users or devices by putting fake code or data into a fake IoT device that has been added to the network.
5. IoT devices are subject to fault injection, differential power analysis, and timing assaults. For these attacks to work, the attacker must have physical access to the device, cause a fault, or passively watch side-channel measures like power and energy consumption [27], electromagnetic radiation, execution time, speed [28], and bandwidth [29]. So, all these vulnerabilities are related to IoT system efficiency and security.

The Fig. 2 show some proposed characteristics that cause the IoT physical layer vulnerabilities in our model, Table 3 lists the formal model requirements that must be specified in our model.

3.2.2 Physical layer attacks

Physical layer security studies fall into three categories [30]. The first is eavesdropping. Illegally receiving or listening to a lawful signal, the transmitter or receiver cannot detect it since

Table 3 The physical layer security requirements

Sec-Req1	Administrator. a person monitoring IoT devices and the network reduces security risks and vulnerabilities
Sec-Req2	Every system has its own password
Sec-Req3	Ensure all of your passwords are strong and unique
Sec-Req4	Checking updates for IoT systems on a regular basis
Sec-Req5	The baseline monitoring of IoT devices allows checking the functionality of any device
Sec-Req6	There is a typical bandwidth for each IoT device (bandwidth_threshold)
Sec-Req7	Check the state of any devices that need less or the same amount of bandwidth as the threshold
Sec–Req8	There is a typical power Consumption for each IoT device (powerConsumption_threshold)
Sec–Req9	Check the state of any devices that need less or the same amount of power Consumption as the threshold
Sec–Req10	Monitoring the state of the speed of devices connected to the IoT system

the eavesdropper is passive. The second is active attacks, like jamming. A jammer disrupts the communication providing a signal to a receiver when a transmitter sends a signal. Jamming prevents the receiver from decoding the legitimate signal. The third type of attack is spoofing, in which a person or program pretends to be someone else, getting access to sensitive information, enter the systems, steal money, or spread malware. Here, we categorize security attacks primarily in IoT physical layer as jamming and MAC spoofing attacks. They are listed below.

- a-jamming attack

Jamming attacks are a type of denial of service (DoS) attack in which hostile nodes intentionally interfere with networks to stop lawful communication. Attackers plant jammers to interfere with wireless networks. Depending on how they plan to attack, jammers can have the same or different abilities as real network nodes. When a transmitter detects a busy wireless medium or a damaged signal received at a receiver, it will back off. A jammer can jam a network in multiple ways for maximum effectiveness. A jammer's function determines whether it's basic or advanced [31]. It divided the elementary jammers into proactive and reactive subgroups. The advanced kinds are function-specific and smart-hybrid. In the past, this method was used for both mobile frequency jamming and RF frequency jamming for WSN nodes (see Fig. 3).

IoT devices employ wireless connections to program, receive user instructions, or transmit data to the cloud. An

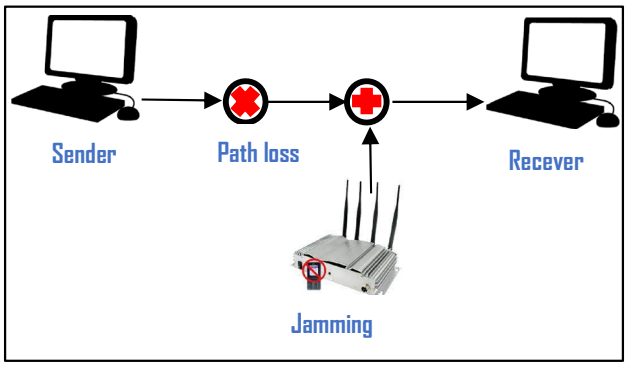

Fig. 3 The jamming attack

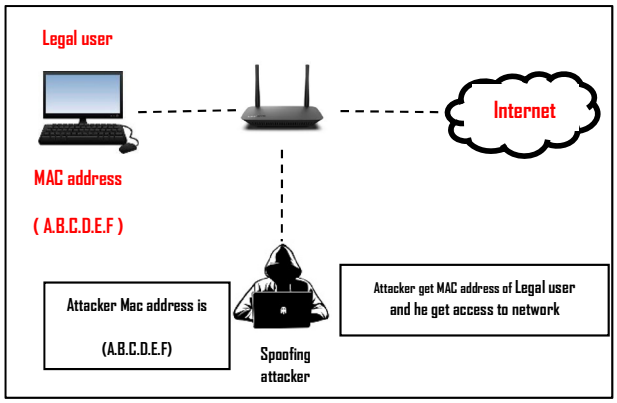

Fig. 4 The MAC spoofing attack

adversary can jam communication between IoT devices and gateways using a frequency that is identical to that of the IoT device. IoT systems are vulnerable to jamming attacks that can cause the batteries of target devices to deplete quickly by disrupting their data transfer and forcing them to retransmit repeatedly [32]. If the attacker believes that there will be a successful transmission, they will jam the data transmission in such a way that the receiver will not be able to decipher the data that has been sent.

- b- spoofing attack

In an IoT access network, identity spoofing attacks can be easily launched. An attacker who uses identity spoofing can pretend to be another real IoT device using the real user's MAC (media access control) or IP (internet protocol) address as a fake ID [33, 34]. Spoofing attackers have an advantage in transmitting the deceptive signal to the receiver because they can do it at higher power. At this time, it is impossible for an attacker to be in the same physical place as a victim. Because the victims' secret keys may have already been compromised by the attackers, cryptography-based authentication is unlikely to be effective. Attackers might be stationary or on the move.

The objective is to fool the recipients. Spoofing can be performed in two situations: (a) when the transmitter stops transmitting the signal, the spoofer can begin transferring a deceiving signal to the receiver; and (b) during the transmission phase between transceivers, the spoofer can transmit the deceiving signal with increased power to the receiver [34]. So, the receiver takes the spoofing signal as a real signal and ignores the real signal coming from the transmitter.

By forging a node's MAC address, a MAC spoofing attack can make it look like a valid node [35]. Mac spoofing is the act of modifying the MAC address on a network interface controller (NIC) card. The MAC address is "burnt in" during manufacturing. Consequently, each network card leaves the factory with a unique MAC address. An attacker can acquire access to data by redirecting a device to another device. In reality, a spoofing attack is a Computer identity theft, is relatively simple. In reality, a spoofing attack is a simple way to steal someone's identity on a computer. This can be done for good or bad reasons (see Fig. 4). Table 4 lists the formal model requirements that must be specified in our model.

Table 4 The physical layer attacks requirements

Jam–Req1	The jammer constantly checks the wireless channel and sends out a signal that interferes with any radio activity it finds
Jam–Req2	Specification of a power threshold to differentiate channel noise from ongoing transmission activity
Jam–Req3	If the estimated received power is higher than the threshold, the jamming signal is sent. If the estimated received power is lower than the threshold, jamming is not happening
MAC-Req1	Each device connected to the network has a MAC address (network card)
MAC-Req2	The attacking device's MAC address must be different from that of the sender and receiver
MAC-Req3	An attacker can get access to data by redirecting a device (destination device) to another device (attacker device)
MAC-Req4	If the attacker's address matches the destination address, spoofing will occur. Otherwise, nothing would occur

3.3 An Event-B method

Event-B is a formal method for discrete system modeling based on the B-Method and developed from the idea of action systems. It's been employed in safety–critical systems [36]. For the correct-by-construction evolution of reactive systems, it depends on first-order logic, typed set theory, and integer arithmetic. The ability to develop correct-by-construction system designs is a significant characteristic of this method. The remaining effort is to develop or implement

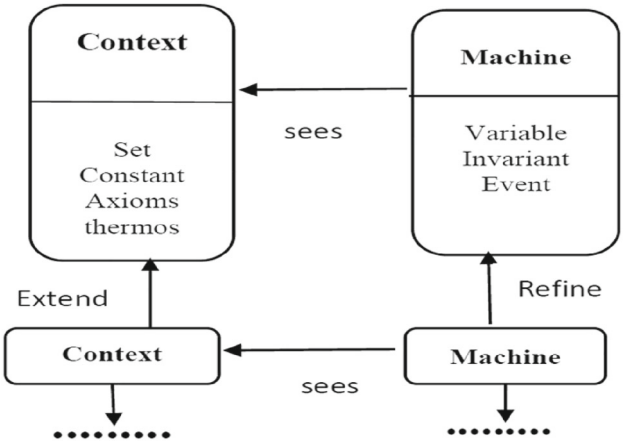

Fig. 5 An Event-B project structure

the right code from the correct design once it has been determined. Event-B is currently centered on the concept of events in general, which is also seen in other formal approaches such as Action Systems [37], TLA [38], and UNITY [39].

In terms of CONTEXT and MACHINE, the Event-B model is made up of static and dynamic parts.

- The CONTEXT FILE: describes the model's static parts. It is a type of first-order theory that provides CONSTANTS declarations that are either *sets* (are types defined by the user) or *numbers*. Constants are declared within a context, and *Axioms* and *Theorems* specify their attributes and connections. An AXIOMS is a statement of a property that cannot be inferred from another axiom. THEOREMS define characteristics that should be derivable from the axioms. The Context structure is:

CONTEXT
< context_name >
EXTENDS
<name of extended contexts >
SETS
< lists of carrier sets>
CONSTANTS
< list the different constants used in this context>
AXIOMS
< label >: < predicate >
. . .
THEOREMS
< label >: < predicate >
. . .
END

-The MACHINE: implements the dynamic part of the model, It uses constants and axioms that are imported from context via the SEES clause to define the dynamic structure or system evolution. The machine must describe: The system's state via a collection of VARIABLES, The state's consistency via a set of INVARIANTS, and, A few EVENTS to specify the machine's probable state evolutions. Every event is expressed in the following format:

ANY *parameters* **WHEN** *guards* **THEN** *actions* **END**.

The guard specifies the required condition for an event to occur. If there is a value for its parameter that causes its guard to hold in this state, the event is said to be enabled in that state. When an event occurs, the action explains how the state variables change. The machine's structure is

MACHINE
< machine's name >
REFINES
< a name of machine that is refine >
SEES
< a name of context that is refine >
VARIABLES
< list of identifier >
INVARIANTS
< label >: < predicate >
. . .
VARIANT
< variant >
EVENTS
< list of event >
END

The key benefit of using Event-B is that it enables models to evolve iteratively through methods such as context EXTENDS and machine REFINEMENT. This is the primary characteristic of Event-B. With the help of these technologies, users can build systems based on their abstract specifications, and they can then add more details about how the systems will work [40]. In other words, a machine can be "*refined*" by another machine, and a context can be "*extended*" by another context. A machine can also "see" a single context or a number of contexts. In Fig. 5, relationships between the machine and the context are shown. Machines and contexts must adhere to the following visibility rules:

4 Proposed model and case studies

4.1 Proposed model

Building a secure, dependable, and error-free IoT system and ensuring its correct operation is a challenging endeavor. In actuality, constructing IoT systems can present a number of challenges. So, for this reason we proposed our model, which

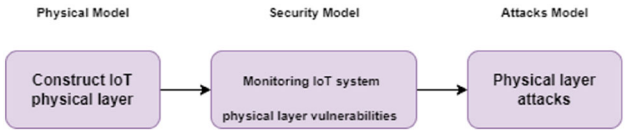

Fig. 6 Proposed model

proposes a new formal model of IoT physical layer security and attacks that is based on Event-B (see Fig. 6):

1. A formal approach to constructing IoT physical layer:
   IoT system contains "Things" which are entities or physical objects that can be uniquely identified, have an embedded system, and share data via a network. Our proposed model requires that all "things" (i.e., entities) have to identify information (such as a name, a type, a location, and properties). IoT parts include things like gadgets, sensors, and motors (each "thing" represents one component). The physical layer's main job is to get information from the environment. We can add sensors and actuators and use information from any IoT device.
2. Security Model:
   We would verify the IoT physical layer vulnerabilities and monitor IoT's system by suggesting a formal approach in several aspects: (a) assign administrator: IoT device and network monitoring by a person or more than one is helpful in lowering security vulnerabilities and risks; (b) use a strong and unique password: make sure each password you use is secure and different; (c) regular check for updates: monitoring the status of software updates for IoT devices regularly;d - monitoring baseline of devices: (i) typical bandwidth: check the functionality of any devices whose bandwidth needs are less than the threshold or are the same as reported by Orsini et al. [41]; (ii) power consumption: check the functionality of any devices whose power consumption needs are less than the threshold or are the same as reported by Prieto et al. [42]; (iii) speed: check the speed of devices that are connected to the IoT system [43].
3. Attacks Model:
   Two types of attacks (jamming and MAC spoofing attacks) are proposed to formally verify the IoT physical layer:
   - The jammer continuously scans the wireless channel and emits a signal that disrupts any radio activity it detects. Specification of a power threshold to distinguish channel noise from ongoing transmissions. If the estimated received power exceeds the threshold, a jamming signal is transmitted. If the estimated received power is below the threshold, there is no interference.
   - MAC addresses are assigned to every network-connected device (network card). The MAC address of the attacking device must be distinct from the sender and receiver. By rerouting a device (destination device) to another device, an attacker can gain access to its data (attacker device). Spoofing will occur if the attacker's address corresponds to the destination address. Aside from that, nothing would occur.

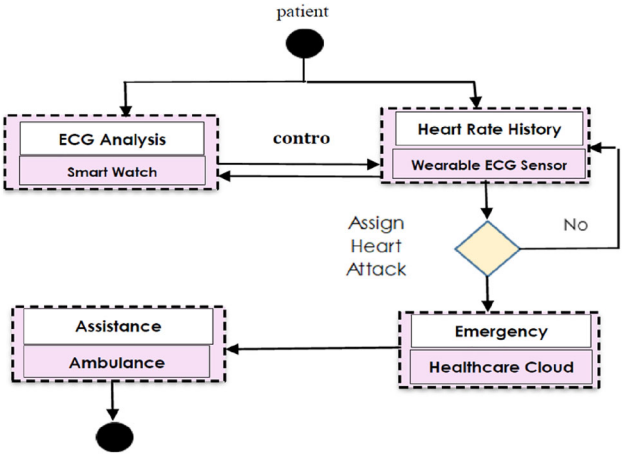

Fig. 7 An overview of the electrocardiogram (ECG) system

4.2 Case studies

To better understand our model, we'll use an IoT system for electrocardiogram (ECG) monitoring in this section. real-time Monitoring of Electrocardiograms (EMoNet) [44]. It's a smart city network made up of patients with heart conditions and a slew of ambulances, smartwatches, and other wearable ECG sensors. EMoNet's flow diagram is shown in Fig. 7, which shows an individual patient's three-minute timed job. It mostly involves getting an electrocardiogram (ECG) and figuring out what it means, as well as calling an ambulance if the patient shows signs of a heart attack.

There are four components to electrocardiogram (ECG) monitoring, and each things in an IoT system reflects one of the following components:

1. Smartwatches (*smartwatch*): provides the ECG Analysis.
2. Wearable ECG Sensor (*Wearable*): provides the Heart Rate History.
3. Healthcare Cloud (*Cloud*): offers emergency service.
4. Ambulance (*Ambulance*): provide Assistance.

The control flow is triggered at regular intervals by ECG Analysis (*smartwatch*), which then passes control to Heart Rate History (*Wearable*) to obtain the most recent sensor reading. Then, ECG Analysis regains control once Heart Rate History hands it back to it. In the event that there are indications of a heart attack, control is transferred from

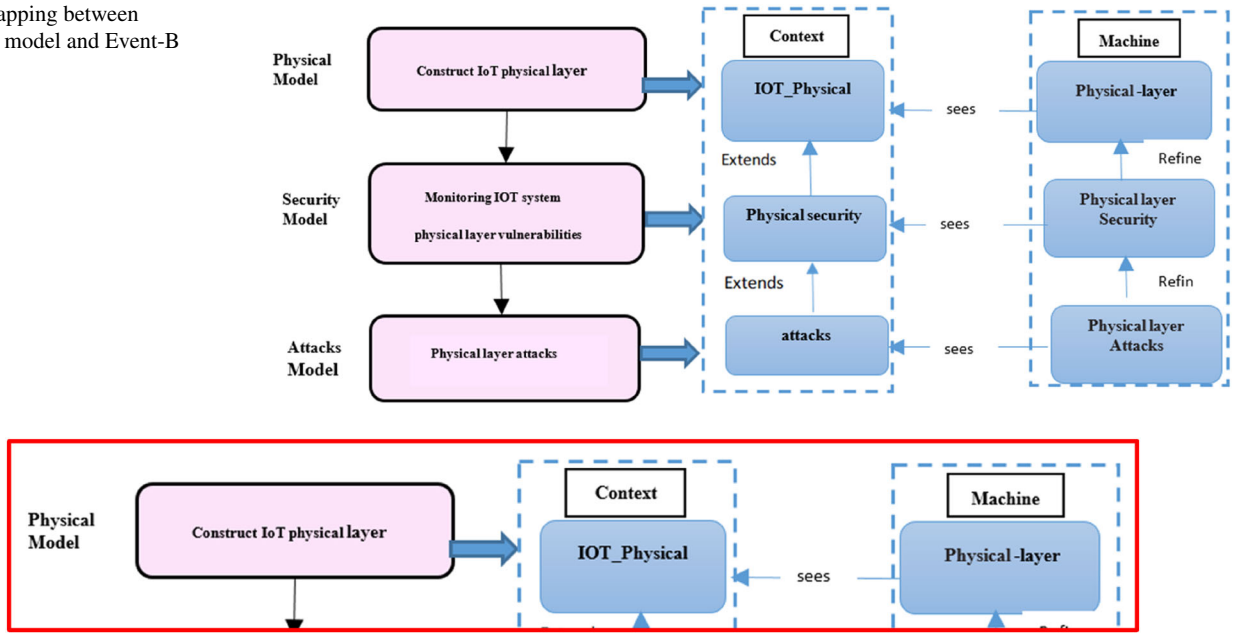

Fig. 8 Mapping between proposed model and Event-B model

Fig. 9 Physical model

ECG Analysis to the emergency service by Healthcare Cloud (*Cloud*), which then transfers control to the assistance service of the closest ambulance (*Ambulance*). After going via the emergency services, control is handed back to the ECG analysis. The physical layer consists of all four components used to represent things [45].

To meet the aforementioned requirements, we present in this study a formal approach to modeling electrocardiogram (ECG) monitoring based on the Event-B method. The Event-B specification, with its formal syntax and semantics, is capable of validating the behavior of created models through the execution of multiple verification features.

To support the generalizability of the proposed method, we test another scenario by checking the IoT system-based fire alarm.

Second Scenario. In the second scenario, there are six components to Fire Alarm System that represent the things in an IoT system:

1. Smoke detecting (*SmokeDet*).
2. Temperature Sensing (*TempSens*).
3. Gas Detection (*GasDet*).
4. Fire Analysis (*FireAnaly*).
5. Alarm Control (*AlarmCont*).
6. Water Sprinkling (*WaterSpr*).

It is begins by collecting data from two types of sensors: smoke detectors (*SmokeDet*) and temperature sensors (*TempSens*). Fire Analysis (*FireAnaly*) will begin its work once it has received real-time data from the preceding sensors. The fire analysis is responsible for analyzing the data received from smoke detection and temperature sensors and comparing it to a predetermined threshold to determine whether or not there is a fire. When a decision has been reached, alarm control (*AlarmCont*) and water sprinklers (*WaterSpr*) are activated. Thus, alarm bells and water sprinklers have been developed simultaneously [46]. In Sect. 6, we show how our model can be used with the two case studies.

5 Proposed model for formalization IoT physical layer in Event-B

In this section, our strategy for formalizing the physical layer of IoT systems in terms of security and attacks is described. The IoT physical layer formal model, as shown in Fig. 8, consists of an abstract model and two refinements. We begin by modeling the physical layer in the abstract model, then refine it to model the physical layer security, and refine it again to represent the physical layer attack.Model three contexts and machines is what we're proposing to do.

5.1 Physical layer structure model

This section describes in detail how the abstract model was constructed. The structure of the physical layer is represented by the abstract model. The primary contribution of this model is a formal way to build the physical layer of an IoT system

Table 5 Specifications of requirements in a model

Phy-Req1	Inv1, Inv2, Inv3, Inv4
Phy-Req2	Inv5, Inv6, Inv7
Phy-Req3	Inv9
Phy-Req4	act1 in Event addData
Phy-Req5	act1 in Events addSensor
Phy-Req6	act1 in Events addActuator
Phy-Req7	act1 in event collectDataSensor

```
CONTEXT
        IOT_physical
SETS
    thing ,nameset,
    typeset, proset,  locatset, SENSOR,
    ACTUATOR, DEVICE,DATA
 AXIOMS
        axm1  :  finite(thing)
        axm2  :  finite(ACTUATOR)
        axm3  :  finite(SENSOR)
END
```

Fig. 10 An Event-B model for IoT architecture: the context (1)

(see Fig. 9). It is made up of the context and the machine, as detailed bellow:

A. Specifications

Here are the IoT physical layer requirements rewritten for the abstract model. The contents of the IoT physical layer are specified by the first three. The following four represent the model events that describe the behavior of this layer as illustrated in Table 5.

B. Context

IOT-physical is the initial context for the IoT physical layer in the Event-B model. Sets (clause SETS) that describe different notions of an abstract model, such as defining IoT things as (thing, thing name, thing type, thing properties, and thing location), are included in the collection. The names of the fields of variables that are relevant to this kind of data are things like nameset, typeset, proset, and locatset. Also, the description of the containing physical layer includes things like sensors, actuators, devices, and data with the set names SENSOR, ACTUATOR, DEVICE, and DATA. The AXIOMS illustrate the characteristics of the qualities whose values were derived from the SETS. In this scenario, the axm1, axm2, and axm3 variables are introduced so as to stipulate that all of the defined sets of things, SENSOR, and ACTUATOR, are, in fact, finite (see Fig. 10).

C. Machine

The"physical layer" machine is in charge of most of the ideas in the physical layer. It is made up of a group of things, each of which has a physical part that is represented by sensors, actuators, and devices (that meet the Phy1 and Phy2

```
MACHINE
        physical layer
SEES
        IOT_physical
VARIABLES
        ..............
INVARIANTS
        inv1  :  thingname ∈ thing ↔ nameset
        inv2  :  thingtype ∈ thing ↔ typeset
        inv3  :  thingpro ∈ thing ↔ proset
        inv4  :  thinglocat ∈ thing ↔ locatset
        inv5  :  sensor ⊆ SENSOR
        inv6  :  actuator ⊆ ACTUATOR
        inv7  :  DEV ⊆ DEVICE
        inv8  :  data ⊆ DATA
        inv9  :  data_sensor ∈ sensor ↔ data
```

Fig. 11 An Event-B model for IoT architecture: the machine (1)

requirements). The gathering of information from the environment of IoT devices is this layer's primary responsibility. In other words, the machine is represented by the clause SEES in the preceding context of IOT-physical, which is the first machine in the Event-B model for the IoT physical layer, as shown in Fig. 11. As a next step, we created a set of variables to represent the abstract model's components. Phy-Req1 is a representation of each thing that has a name, a type, properties, and a location, and the invariants inv1, inv2, inv3, and inv4 describe this. Phy-Req2 is modeled using the following: inv5, inv6, and inv7. The data variable is defined as a subset of the data set by this machine (invariants inv8). This is followed by the model of the sensor Phy-Req3 by the invariant inv9.

The first event in the abstract model, the addData event, is used to add data from any IoT device that meets Phy-Req4 and is included in the data set (grd1) but not in the data set (grd2), and then added to the data set (act1). In a manner analogous to this event, the addSensor and addActuator events include the addition of one or more sensors and actuators, respectively, that satisfy Phy-Req5, and Phy-Req6 as shown in Fig. 12.

In the physical layer of an IoT structure, sensors or other devices are used to gather data from the IoT environment. The collectDataSensor event tells us what this function is, and Phy-Req7 is modeled to match (see Fig. 13).

5.2 Physical layer security model

In this model, we'd check for any IoT physical layer vulnerabilities, such as weak passwords and outdated systems, and we'd examine the baseline (speed, normal bandwidth, and

```
addData
 STATUS
    ordinary
ANY
  d
WHERE
     grd1 : d ∈ DATA
     grd2 : d ∉ data
THEN
     act1: data ≔ data ∪ {d}
END
addSensor
STATUS
     ordinary
   ANY
   s
  WHERE
     grd1: s ∈ SENSOR
     grd2: s ∉ sensor
  THEN
     act1: sensor ≔ sensor ∪ {s}
END
addActuator
 STATUS
ordinary
  ANY
  ac
 WHERE
     grd1: ac ∈ ACTUATOR
     grd2: ac ∉ actuator
 THEN
     act1: actuator ≔ actuator ∪ {ac}
END
```

Fig. 12 The events of adding data, sensor, and actuator for IoT architecture

```
collectDataSensor   ≙
STATUS
 ordinary
ANY
 d
 s
WHERE
 grd1   :   d ∈ data
 grd2   :   s ∈ sensor
 grd3   :   s∉dom(data_sensor)∧d∉ran(data_sensor)
THEN
 act1   :   data_sensor ≔ data_sensor ∪ {s ↦ d}
END

END
```

Fig. 13 The collectDataSensor event for IoT architecture

power consumption) to make sure there aren't any. Because they are used in conjunction with security-sensitive functions on the device, IoT physical layer vulnerabilities hold a special place in the formal verification process. Complete device testing and attacks detection are therefore made more challenging. During Event-B, the Physical layer is brought in by a refining of the process (see Fig. 14). Therefore, we will develop a second level of abstraction that will further refine the first level.As illustrated below:

Table 6 Specifications of requirements in a model

Sec-Req1	Inv1, Inv3
Sec-Req2	Inv4, Inv5
Sec-Req3	Inv6, Inv7, grd1,grd2,grd3 in (Event passwordChecking)
Sec-Req4	Inv8, Inv9, grd1,grd3 and act1 in (Event updateChecking)
Sec-Req5	That satisfied in all Specifications bellow
Sec-Req6	Inv11, Inv12, Inv13
Sec-Req7	grd1, grd2, grd3, act1 in (Event bandwidthMonitoring)
Sec–Req8	Inv15, Inv17, Inv18
Sec–Req9	grd1,grd2, grd3, act1 in (Event powerMonitoring)
Sec–Req10	Inv19, Inv20 and grd1 grd2,act1 in (Event speedMonitoring)

A. Specifications:

The following are rewritten requirements for the first refinement to describe some of the IoT's physical layer vulnerabilities that cause insecurity to the system. As illustrated in Table 6.

B. First Refinement:

In this section, we extend the structure of the physical layer to the formalization we made in the previous section. So, that physical layer security can be shown in a formal way using the refinement concept of Event-B by adding a variable and five new events: *updateChecking, passwordChecking, bandwidthMonitoring, powerMonitoring, and speedMonitoring*, (see Fig. 15).

According to our proposed for this model, we illustrated the more effective vulnerabilities in IoT physical layer.

1. 1-Put someone in charge of managing everything. Having a person who acts as the administrator of all of the devices connected to the internet of things as well as

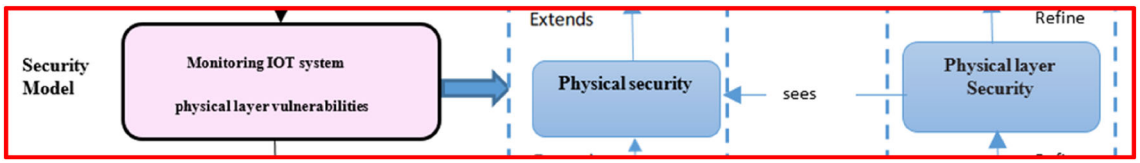

Fig. 14 Security model

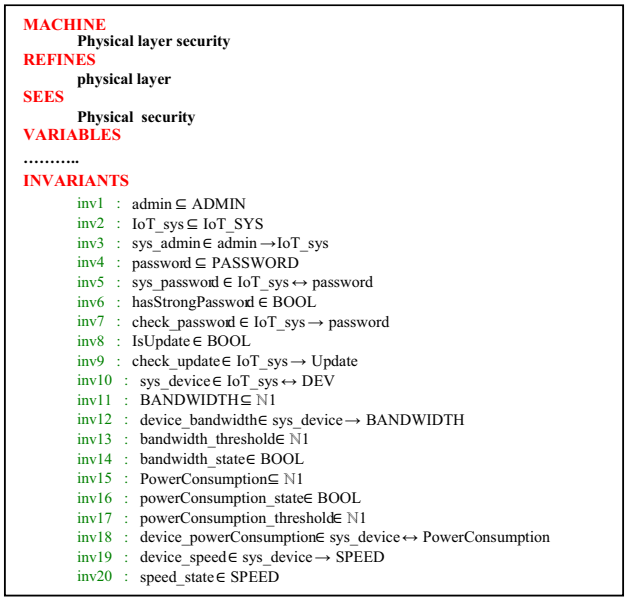

Fig. 15 An Event-B model for IoT physical layer security: the machine (1)

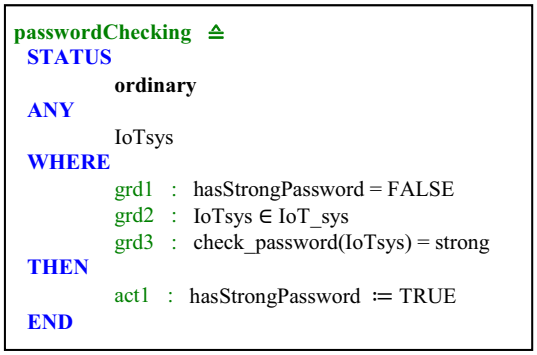

Fig. 16 The passwordChecking event for physical layer security

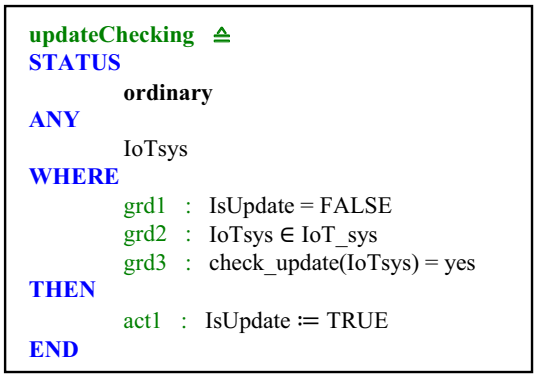

Fig. 17 The updateChecking event for physical layer security

the network can help reduce the number of security oversights and vulnerabilities. In our model, the administrator is represented by the variable *admin*, which is shown as the invariants inv1 and inv3. It is set up to assign a system administrator that satisfies **Sec-Req1**.

2. 2-Make sure all of your passwords are strong and unique. Many cyberattacks can be thwarted with the use of strong passwords. A password manager is a piece of software that makes it easier for users to set secure passwords and keep them safe within the application they use, this is modeled **Sec-Req2** and **Sec–Req3**.
3. The invariants (inv4 and inv5) serve to specify the system password. And, we suggest adding a Checking password to check if it is strong or weak password. This is modeled by the event *passwordChecking* as shown in Fig. 16 where a component *hasStrongPassword* is checked by another variable *check_password* (Guards grd1, grd2, and grd3).
4. 3-Check updates on a regular basis. The Internet of Things (IoT) is plagued by numerous security flaws. This is due to the fact that IoT devices can be vulnerable at any level. Cybercriminals are still using old flaws to infect devices, which shows how long devices can stay online without being fixed. This part is modeled by **Sec-Req4**. Variable *check_update* and *IsUpdate* are defined in invariants inv8 and inv9, respectively. which represents an enablement of the specification of the system's update.So, An event with the name *updateChecking* is created to model the system's checking for updates (set grd1 and grd3) and ensuring that they are present (set act1) (see Fig. 17)
5. 4-Monitor the device's baseline operation which satisfied **Sec-Req5**. It might be difficult to identify cyberattacks. The baseline behavior (speed, typical bandwidth, power consumption, etc.) of devices might assist users in spotting changes that may indicate malware infections. In this model, we suggest to adding a variable and three new events to monitoring the device behavior: *bandwidthMonitoring, powerMonitoring, and speedMonitoring.*

- Each IoT device has a standard bandwidth barrier the(bandwidth threshold). The invariants inv11, inv12 and inv13 define how to model the bandwidth (*BANDWIDTH*). It must be a number, and each device has to have it (*device_bandwidth*) and have a threshold (*bandwidth_threshold*), respectively which models the requirement **Sec–Req6**. The event (*bandwidthMonitoring*) is designed to model the monitoring of the state bandwidth of any devices that require less or the same amount of bandwidth as the threshold set (grd1, grd2, and grd3), and to decide the state (act1) that satisfies **Sec-Req7** (see Fig. 18).

- Power consumption limits the battery life of portable devices like cell phones and laptop computers. For each IoT

```
bandwidthMonitoring ≙
STATUS
        ordinary
ANY
        IoTsys
        IoT_device
        bandwidth
WHERE
        grd1  :  bandwidth_state = FALSE
        grd2  :  bandwidth ∈ ran(device_bandwidth)
        grd3  :  bandwidth ≥ bandwidth_threshold
        .........
THEN
        act1  :  bandwidth_state ≔ TRUE
END
```

Fig. 18 The bandwidthMonitoring event for physical layer security

```
powerMonitoring ≙
 STATUS
        ordinary
 ANY
        IoTsys
        IoT_device
        power_consumption
 WHERE
   grd1  :  powerConsumption_state = FALSE
   grd2  :  power_consumption ∈ PowerConsumption ∧ power_consumption
            ∈ ran(device_powerConsumption)
   grd3  :    power_consumption ≥ powerConsumption_threshold
   ..........
 THEN
        act1  :  powerConsumption_state ≔ TRUE
 END
```

Fig. 19 The powerMonitoring event for physical layer security

device has a standard power consumption, set (inv17), which models the requirement **Sec–Req6**, and the (inv15 and inv18) define the power consumption of device. (grd1, grd2, and grd3) are meant to model the monitoring of state power consumption in any devices that need less or the same amount of power as the threshold set by the event (*powerMonitoring*) and to decide the state (act1) that meets Sec-Req9 (see Fig. 19).

- Devices connected to the IoT network can be monitored for their current speed. The (inv19 and inv20) models the requirement **Sec–Req10** regarding the device speed (*device_speed*), and device state (*speed_state*). The (*speedMonitoring*) event is utilized to monitor the speed status of the device. Every IoT device has a speed within a particular range set (grd1 and grd2), then the speed state is checked (act1), as shown below (Fig. 20).

```
speedMonitoring ≙
 STATUS
        ordinary
 ANY
        IoTsys
        IoT_device
        speed
 WHERE
    grd1  :  (IoTsys ↦ IoT_device) ∈ dom(device_speed)
    grd2  :  speed ∈ SPEED  ∧ speed ∈ ran(device_speed)
 THEN
    act1  :  speed_state ≔ device_speed(IoTsys ↦ IoT_device)
 END
```

Fig. 20 The speedMonitoring event for physical layer security

5.3 Physical layer attacks model

In this model, we used jammer and MAC spoofing attacks as our models for the more significant IoT physical layer attacks, as shown below (Fig. 21).

A. Specifications:

The following requirements for the second refinement have been rearranged to describe some of the attacks on the system that use the IoT's physical layer according to our model. Table 7 shows a structure that can be used to get physical layer attack parameters from the proposed method.

B. Second Refinement

This part shows the second refinement to the IoT physical layer model, which is attacks on the physical layer. This refinement adds the physical layer representations for the jamming and MAC spoofing attacks to the formalization that was made for the abstract model in the last refinement. That is illustrated in Fig. 22.

- Jamming attack: An IoT device and a gateway can not talk to each other if an attacker with a jammer uses the same frequency as the IoT device. For this purpose, we model how a jammer operates in the formalization method.

The jammer is always looking for radioactivity on the wireless channel and sending out a signal to stop it, this modeled **Jam–Req1**. The inv1, inv2 define jamming attack (*DOSjammer_attack*) and send a jamming signal (*send_DOSjammer_signal*), respectively. And the event (*DOSJammingAttack)* check this signal in grd2. After that, The specification of a power threshold (*power_ threshold*) set (inv5), received power (*POWER_RECEIVED*) set (inv3), and the channel communication power (*communication_power*) set (inv4) is required to distinguish channel noise from continuous transmission activity (grd3 and grd4) which models

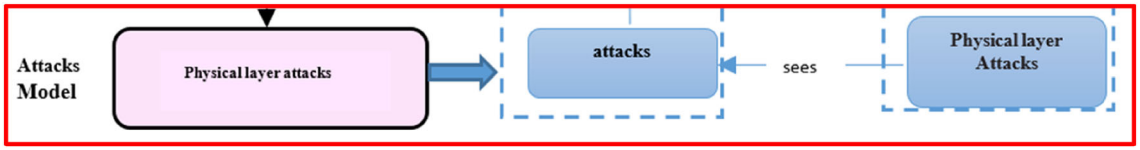

Fig. 21 Attacks model

Table 7 The physical layer Attacks requirements

Jam–Req1	inv1, inv2,inv3,inv4 and grd2 in event (DOSJammingAttack)
Jam–Req2	inv5,grd3, grd4
Jam–Req3	grd5, act1
MAC-Req1	inv14, inv15, event (MACSpoofingAttackChecking)
MAC-Req2	inv16
MAC-Req3	Inv17,grd2, grd3
MAC-Req4	grd4, act1

MACHINE
Physical layer Attacks
REFINES
Physical layer security
SEES
Attacks
VARIABLES
.....
INVARIANTS
inv1 : DOSjammer_attack ∈ ATTACK
inv2 : send_DOSjammer_signal ∈ BOOL
inv3 : POWER_RECIEVED ⊆ N1
inv4 : communication_power ∈ COMMUNICATION_CHANNEL ↔ POWER_RECIEVED
inv5 : power_threshold ∈ N1
inv6 : IsSecureJamming ∈ BOOL
inv7 : IsSecureMACSpoofing ∈ BOOL
inv8 : SENDER_DEV ⊆ DEV
inv9 : RECIEVER_DEV ⊆ DEV
inv10 : ATTACK_DEV ⊆ DEV
inv11 : senderMACAddress ⊆ MAC_ADDRESS
inv12 : recieverMACAddress ⊆ MAC_ADDRESS
inv13 : attackMACAddress ⊆ MAC_ADDRESS
inv14 : sender_MAC ∈ SENDER_DEV ↔ senderMACAddress
inv15 : reciever_MAC ∈ RECIEVER_DEV → recieverMACAddress
inv16 : attack_MAC ∈ ATTACK_DEV → attackMACAddress
inv17 : send_data ∈ DATA → MAC_ADDRESS

Fig. 22 An Event-B model for IoT physical layer attacks: the machine (3)

DOSJammingAttack ≙
STATUS
ordinary
ANY
communication_channel
power_recieved
WHERE
grd1 : IsSecureJamming = TRUE
grd2 : send_DOSjammer_signal = FALSE
grd3 : communication_channel ∈ COMMUNICATION_CHANNEL ∧ communication_channel ∈ dom(communication_power)
grd4 : power_recieved ∈ POWER_RECIEVED ∧ power_recieved ∈ ran(communication_power)
grd5 : power_recieved ≥ power_threshold
THEN
act1 : send_DOSjammer_signal ≔ TRUE
act2 : IsSecureJamming ≔ FALSE
END

Fig. 23 The DOSJammingAttack event for physical layer Attacks

MACSpoofingAttackChecking ≙
STATUS
ordinary
ANY
distination_address
current_address
WHERE
grd1 : IsSecureMACSpoofing = TRUE
grd2 : distination_address ∈ ran(reciever_MAC)
grd3 : current_address ∈ ran(send_data)
grd4 : distination_address ≠ current_address
THEN
act1 : IsSecureMACSpoofing ≔ FALSE
END

Fig. 24 The MACSpoofingAttackChecking event for physical layer attacks

the requirement **Jam–Req2**. So, if the estimated received power exceeds the predetermined threshold (grd5), the jamming signal is transmitted (act1). Jamming does not occur if the estimated received power is less than the threshold; this satisfies (Jam–Req3) (see Fig. 23).

-**MAC spoofing attack:** MAC address spoofing occurs when an attacker changes a network device's MAC address to something else. In our model, we are formally modeling this attack's method and workflow.

To model **MAC-Req1**, we used the invariants (inv14 and inv15) to define the sender's and receiver's MAC addresses. And, the (inv16) defines the MAC address for attack (*attackMACAddress*) which has to be different from the preferences of both the sender and the receiver **MAC-Req2**. However, an attacker can acquire access to a data set (inv17). The event (*MACSpoofingAttackChecking*) covers MAC Req3 by redirecting a device's destination (*distination_address*) to a different device (*current_address*) set (grd2, grd3). Spoofing will occur if the attacker's address matches the destination address (grd4). If not, nothing would happen. Set the act1 and MAC-Req4 requirements (see Fig. 24).

Fig. 25 The proof obligations of IoT physical layer formal model

6 Verification and validation

In this section, the steps taken to verify and validate our model are described. The Rodin platform is used to model and validate Event-B for model validation. Although some difficult proof obligations (POs) [47] necessitate manual interactive verification, most of the proof obligations can be proved automatically using the Rodin platform, which automatically creates the proof obligations for each theorem and invariant. The accuracy generation invariants must meet their proof obligations at each level of the model. Validation is accomplished through close observation of the specification's operation. A plugin called ProB is available on the Rodin platform for animating and validating Event-B requirements.

Element name	Total	Auto.	Man.	Rev.	Und.
IOT architecture	**89**	**89**	**0**	**0**	**0**
physical layer	19	19	0	0	0
Physical layer security	35	35	0	0	0
Physical layer Attacks	30	30	0	0	0
IOT_physical	0	0	0	0	0
Physical security	0	0	0	0	0
Attacks	0	0	0	0	0

Fig. 26 The proof statistics of IoT physical layer formal model

6.1 Proof-based verification

The Proof Obligation Generator is a feature of the Rodin Platform that generates proof obligations (POs) mechanically. In general, proof tasks were either well-definedness (WD) or invariant preservation (INV) types (for the model's invariants). The invariant preservation operators (INV POs) guarantee that all events comply with the specification.

The purpose of the well-definedness proof obligation rule (WD) is to guarantee that a potentially ill-defined theorem, axiom, invariant, guard, action, witness, or variant is, in fact, well defined. The following element names are used when referring to a particular modeling component: inv/WD, thm/WD, axm/WD, grd/WD, act/WD, VWD, and evt/x/WWD. The way this proof obligation rule is put into action depends on the expression, which may or may not be clear. A PO may be discharged either automatically or interactively (indicated by the green symbol), or it may remain undischarged (orange symbol). When you see the letter "A" on the PO, it indicates that the discharge is processed automatically.

For example, in our model, the invariant (INV1) in the abstract model (physical layer)*inv1: thingname* $\in$ *thing* $\leftrightarrow$ *nameset*when verifying this invariant by proof obligation, if assigned a name to things, then it is discharged and indicated by (green symbol), if not, it remains undischarged (orange symbol).

In our model, (see Fig. 25); 100% of proof obligations were automatically satisfied by the Rodin prover. The complexity of the proof obligation is correlated with the speed at which automatic proofs can be completed. The robust Rodin

plugins greatly facilitate our work by automating the majority of the routine mechanical tasks. But careful modeling and smarter automatic proof tools can cut down on the amount of time people have to spend on verification tasks. Figure 26 shows the proof statistics that illustrates the distribution of automatically and manually discharged POs.

It's important to note that the Event-B approach does not have the state explosion problem [48], which is a problem with other verification techniques used in the literature.

6.2 Validation by ProB model checker

There are numerous potent plugins for Rodin. We use the ProB animation tool [47] in our work. For the Event-B approach, ProB is a constraint solver and model verifier that enables both the validation of the requirements and the detection of errors to fix them. It is not necessary for the user to intervene to demonstrate most of the proof obligations. Using ProB's constraint-solving capabilities, you may create models, check for deadlocks-free, and generate test cases. It needs a concrete model to be done, and cannot be done on an abstract Event-B specification. We were able to perform an automated animation to verify the accuracy of the model specification step-by-step using an animator checker tool. We were able to bring the case studies to life in our model by altering the values of its many variables, carriers, and constants. The initialization parameters of an electrocardiogram (ECG) IoT system shown in Fig. 27 must be used to put this animation into action. To support the generalizability of the proposed method using another case study fire alarm system, see Fig. 28, which shows the initialization parameters of this case study. By monitoring the various states, we may determine whether the model behaves accurately. ProB tested the model's events and found no problems at any level. This shows that the model's behavior has been fixed.

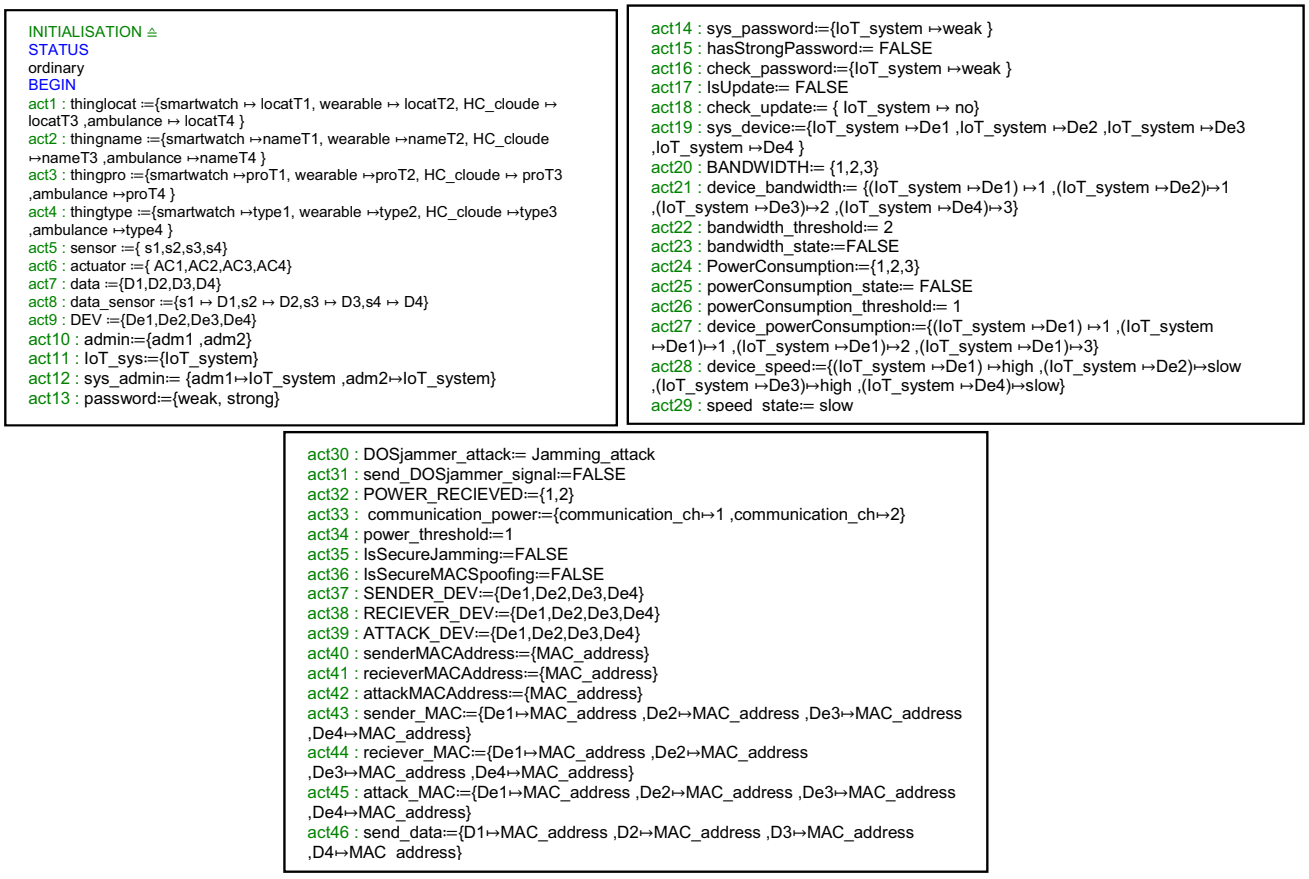

Fig. 27 Initialization values of the electrocardiogram (ECG) system

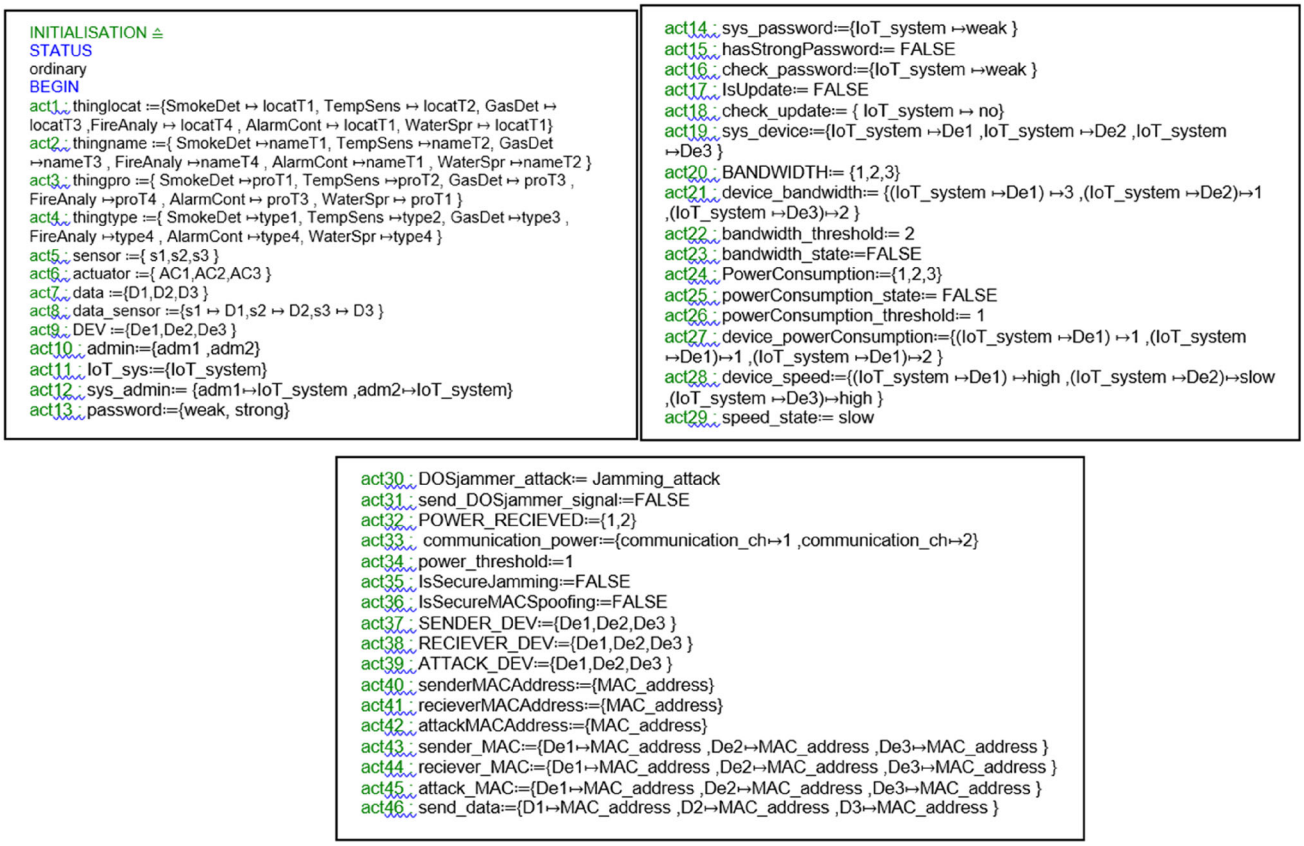

Fig. 28 Initialization values of the fire alarm system

7 Conclusion

In this research, we present a method for verifying and validating the security and attacks of the IoT's physical layer. We formalized and validated the physical-layer vulnerabilities and attacks. First, we defined the needs for modeling the IoT physical layer structure, vulnerabilities (by analyzing the absence of changes in a number of characteristics that contribute to these vulnerabilities, such as speed, usual bandwidth, and power consumption), and attacks (jamming and MAC spoofing). Second, an Event-B model was shown to model and test the security and attacks on the physical layer of the IoT using the mechanism for refinements. The refinements made it possible for us to make corrections by constructing all the models and facilitating the proofs. To show how well our method works, we turned our ideas into a real-world example in the form of an electrocardiogram (ECG) IoT system. As a second case study to show how flexible the proposed method is, we looked at the fire alert system. An animator checker called ProB and proof obligations are used to ensure that our formal model is correct. In future work, we intend to address issues that must be resolved for the IoT physical layer's continued security. We would like to address the security of physical layer communication protocols as a software component, as our model only addressed the physical component. Additionally, we intend to address security at the IoT system's other layers.

Data availability No datasets were generated or analyzed during the current study.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Ning H, Wang Z (2011) Future internet of things architecture: like mankind neural system or social organization framework? IEEE Commun Lett 15(4):461–463
2. López TS, Ranasinghe DC, Patkai B, McFarlane D (2011) Taxonomy, technology and applications of smart objects. Inf Syst Front 13(2):281–300
3. Ghosh A, Chakraborty D, Law A (2018) Artificial intelligence in Internet of things. CAAI Trans Intell Technol 3(4):208–218
4. Rizvi S, Pipetti R, McIntyre N, Todd J, Williams I (2020) Threat model for securing internet of things (IoT) network at device-level. Internet Things 11:100240

5. Hamamreh JM, Furqan HM, Arslan H (2018) Classifications and applications of physical layer security techniques for confidentiality: a comprehensive survey. IEEE Commun Surv Tutor 21(2):1773–1828
6. Shakiba-Herfeh M, Chorti A, Poor HV (2021) Physical layer security: authentication, integrity, and confidentiality. In: Physical layer security. Springer, Cham, pp 129–150
7. Wang D, Bai B, Lei K, Zhao W, Yang Y, Han Z (2019) Enhancing information security via physical layer approaches in heterogeneous IoT with multiple access mobile edge computing in smart city. IEEE Access 7:54508–54521
8. Alladi T, Chamola V, Sikdar B, Choo KKR (2020) Consumer IoT: security vulnerability case studies and solutions. IEEE Consumer Electron Mag 9(2):17–25
9. Wang N, Wang P, Alipour-Fanid A, Jiao L, Zeng K (2019) Physical-layer security of 5G wireless networks for IoT: challenges and opportunities. IEEE Internet Things J 6(5):8169–8181
10. Ullah F, Al-Turjman F, Nayyar A (2020) IoT-based green city architecture using secured and sustainable android services. Environ Technol Innov 20:101091
11. Keerthi K, Roy I, Hazra A, Rebeiro C (2019) Formal verification for security in IoT devices. Security and Fault Tolerance in Internet of Things, pp 179–200
12. Bae WS (2019) Verifying a secure authentication protocol for IoT medical devices. Clust Comput 22(1):1985–1990
13. Desnitsky V, Kotenko I (2016) Automated design, verification and testing of secure systems with embedded devices based on elicitation of expert knowledge. J Ambient Intell Humaniz Comput 7(5):705–719
14. Kammüller F (2017) Formal modeling and analysis with humans in infrastructures for iot health care systems. In: International conference on human aspects of information security, privacy, and trust. Springer, Cham, pp 339–352
15. Kammüller F (2017) Human centric security and privacy for the iot using formal techniques. In: International conference on applied human factors and ergonomics. Springer, Cham, pp 106–116
16. Dhillon PK, Kalra S (2017) Secure multi-factor remote user authentication scheme for Internet of Things environments. Int J Commun Syst 30(16):e3323
17. Drozdov D, Patil S, Dubinin V, Vyatkin V (2017) Towards formal verification for cyber-physically agnostic software: a case study. In: IECON 2017–43rd annual conference of the IEEE industrial electronics society. IEEE, pp 5509–5514
18. Kim H, Kang E, Lee EA, Broman D (2017) A toolkit for construction of authorization service infrastructure for the internet of things. In: Proceedings of the second international conference on Internet-of-Things design and implementation, pp 147–158
19. Mohsin M, Sardar MU, Hasan O, Anwar Z (2017) IoTRiskAnalyzer: a probabilistic model checking based framework for formal risk analytics of the Internet of Things. IEEE Access 5:5494–5505
20. Kars P (1998) Formal methods in the design of a storm surge barrier control system. In: Lectures on embedded systems, European educational forum, school on embedded systems. Springer, London, pp 353–367
21. Zahra S, Alam M, Javaid Q, Wahid A, Javaid N, Malik SUR, Khan MK (2017) Fog computing over IoT: a secure deployment and formal verification. IEEE Access 5:27132–27144
22. Gubbi J, Buyya R, Marusic S, Palaniswami M (2013) Internet of Things(IoT): a vision, architectural elements, and future directions. Futur Gener Comput Syst 29(7):1645–1660
23. Weyrich M, Ebert C (2015) Reference architectures for the internet of things. IEEE Softw 33(1):112–116
24. Zhang AL (2016) Research on the architecture of internet of things applied in coal mine. In: 2016 International conference on information system and Artificial Intelligence (ISAI). IEEE, pp 21–23
25. Zhang N, Fang X, Wang Y, Wu S, Wu H, Kar D, Zhang H (2020) Physical-layer authentication for internet of things via wfrft-based gaussian tag embedding. IEEE Internet Things J 7(9):9001–9010
26. Li C, Palanisamy B (2018) Privacy in internet of things: from principles to technologies. IEEE Internet Things J 6(1):488–505
27. Ali B, Awad AI (2018) Cyber and physical security vulnerability assessment for IoT-based smart homes. Sensors 18(3):817
28. Li X, Luo C, Ji H, Zhuang Y, Zhang H, Leung VC (2020) Energy consumption optimization for self-powered IoT networks with non-orthogonal multiple access. Int J Commun Syst 33(1):e4174
29. Debroy S, Samanta P, Bashir A, Chatterjee M (2019) SpEED-IoT: spectrum aware energy efficient routing for device-to-device IoT communication. Futur Gener Comput Syst 93(833–848):4
30. Xu T, Darwazeh I (2018) Non-orthogonal narrowband Internet of Things: a design for saving bandwidth and doubling the number of connected devices. IEEE Internet Things J 5(3):2120–2129
31. Kamel SOM, Hegazi NH (2018) A proposed model of IoT security management system based on a study of internet of things (IoT) security. Int J Sci Eng Res 9(9):1227–1244
32. Greco C, Pace P, Basagni S, Fortino G (2021) Jamming detection at the edge of drone networks using Multi-layer Perceptrons and Decision Trees. Appl Soft Comput 111:107806
33. Chi Z, Li Y, Liu X, Wang W, Yao Y, Zhu T, Zhang Y (2020) Countering cross-technology jamming attack. In: Proceedings of the 13th ACM conference on security and privacy in wireless and mobile networks, pp 99–110
34. Yousefnezhad N, Madhikermi M, Främling K (2018) Medi: measurement-based device identification framework for internet of things. In: 2018 IEEE 16th international conference on industrial informatics (INDIN). IEEE, pp 95–100
35. Boulkenafet Z, Komulainen J, Hadid A (2018) On the generalization of color texture-based face anti-spoofing. Image Vis Comput 77:1–9
36. Farhin F, Sultana I, Islam N, Kaiser MS, Rahman MS, Mahmud M (2020) Attack detection in internet of things using software defined network and fuzzy neural network. In: 2020 joint 9th international conference on informatics, electronics & vision (ICIEV) and 2020 4th international conference on imaging, vision & pattern recognition (icIVPR). IEEE, pp 1–6
37. Hoang TS (2013) An introduction to the Event-B modelling method. Ind Deploy Syst Eng Methods 211–236
38. Craigen D (1999) Formal methods adoption: what's working, what's not! In: Proceedings of the 5th and 6th international SPIN workshops on theoretical and practical aspects of SPIN model checking. Springer, London, pp 77–91
39. Eisner C (2002) Using symbolic CTL model checking to verify the railway stations of Hoorn- Kersenboogerd and Heerhugowaard. Int J Softw Tools Technol Transf 4(1):107–124
40. Damchoom K, Butler M, Abrial JR (2008) Modelling and proof of a tree-structured file system in Event-B and Rodin. In: International conference on formal engineering methods. Springer, Berlin, Heidelberg, pp 25–44
41. Orsini G, Posdorfer W, Lamersdorf W (2021) Saving bandwidth and energy of mobile and IoT devices with link predictions. J Ambient Intell Humaniz Comput 12(8):8229–8240
42. Prieto MD, Martínez B, Monton M, Guillen IV, Guillen XV, Moreno JA (2014) Balancing power consumption in IoT devices by using variable packet size. In: 2014 eighth international conference on complex, intelligent and software intensive systems. IEEE, pp 170–176
43. Aravindh G, Kowshik A. Speed detection using IOT. Int J Comput Appl 975:8887
44. Muankid A, Ketcham M (2019) The real-time electrocardiogram signal monitoring system in wireless sensor network. Int J Online Biomed Eng 15(2)

45. Gusev M, Poposka L, Spasevski G, Kostoska M, Koteska B, Simjanoska M, Trontelj J (2020) Noninvasive glucose measurement using machine learning and neural network methods and correlation with heart rate variability. J Sensors 2020
46. Georgiades G, Papageorgiou XS, Loizou SG (2019) Integrated forest monitoring system for early fire detection and assessment. In: 2019 6th international conference on control, decision and information technologies (CoDIT). IEEE, pp 1817–1822
47. Leuschel M, Butler M (2013) ProB: A model checker for B. In: International symposium of formal methods Europe. Springer, Berlin, Heidelberg, pp 855–874
48. Ait-Ameur Y, Baron M, Kamel N, Mota JM (2009) Encoding a process algebra using the Event B method: application to the validation of human–computer interactions. Int J Softw Tools Technol Transfer 11:239–253
49. Abrial JR, Butler M, Hallerstede S, Hoang TS, Mehta F, Voisin L (2010) Rodin: an open toolset for modelling and reasoning in Event-B. Int J Softw Tools Technol Transfer 12(6):447–466