


Article

# A Lightweight Continuous Authentication Protocol for the Internet of Things

Yo-Hsuan Chuang <sup>1</sup>, Nai-Wei Lo <sup>1</sup> , Cheng-Ying Yang <sup>2,\*</sup> and Ssu-Wei Tang <sup>1</sup>

<sup>1</sup> Department of Information Management, National Taiwan University of Science and Technology, Taipei 10607, Taiwan; D10009103@mail.ntust.edu.tw (Y.-H.C.); nwlo@cs.ntust.edu.tw (N.-W.L.); M10209101@mail.ntust.edu.tw (S.-W.T.)

<sup>2</sup> Department of Computer Science, University of Taipei, Taipei 10048, Taiwan

\* Correspondence: cyang@utapei.edu.tw

Received: 11 February 2018; Accepted: 3 April 2018; Published: 5 April 2018



**Abstract:** Modern societies are moving toward an information-oriented environment. To gather and utilize information around people's modern life, tiny devices with all kinds of sensing devices and various sizes of gateways need to be deployed and connected with each other through the Internet or proxy-based wireless sensor networks (WSNs). Within this kind of Internet of Things (IoT) environment, how to authenticate each other between two communicating devices is a fundamental security issue. As a lot of IoT devices are powered by batteries and they need to transmit sensed data periodically, it is necessary for IoT devices to adopt a lightweight authentication protocol to reduce their energy consumption when a device wants to authenticate and transmit data to its targeted peer. In this paper, a lightweight continuous authentication protocol for sensing devices and gateway devices in general IoT environments is introduced. The concept of valid authentication time period is proposed to enhance robustness of authentication between IoT devices. To construct the proposed lightweight continuous authentication protocol, token technique and dynamic features of IoT devices are adopted in order to reach the design goals: the reduction of time consumption for consecutive authentications and energy saving for authenticating devices through by reducing the computation complexity during session establishment of continuous authentication. Security analysis is conducted to evaluate security strength of the proposed protocol. In addition, performance analysis has shown the proposed protocol is a strong competitor among existing protocols for device-to-device authentication in IoT environments.

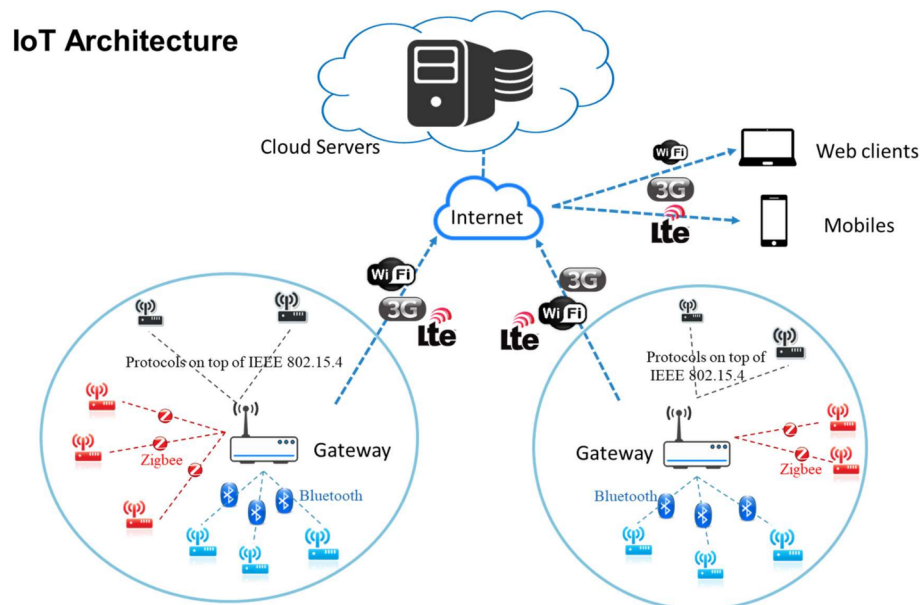
**Keywords:** continuous authentication; device-to-device authentication; token technique; dynamic device feature; Internet of Things

## 1. Introduction

As megacities have emerged rapidly around the world in recent years, how to support all kinds of activities generated as a result of the huge city populations has become an important and practical issue to government agencies. The visions of smart homes, smart buildings and smart cities are some of the promising solutions possible by generating and utilizing new information from cities themselves to intelligently lift up the support level and range of city governments. To realize this vision, various sensing devices and gateways have to be massively deployed so they can collect different new information through their sensing devices and deliver this information to intelligent backend application systems to produce and support value-added services [1] within city environments. To effectively connect sensing devices, intermediary gateways, backend application servers and client devices such as smartphones, tablets and smart watches [2] for data exchange and information delivery, new infrastructure is required. The concept of the Internet of Things (IoT) [3] indicating the huge

number of “things” (objects) which are interconnected through the Internet is suitable to be adopted and defines the new infrastructure as an IoT infrastructure. With deployed IoT infrastructure, a lot of intelligent management services and information synthesis-oriented services can be developed and implemented in various industry sectors such as smart healthcare, intelligent transportation, factory energy management and home appliance control [4,5].

The development and deployment of IoT infrastructure is in its infancy. The generic architecture of the Internet of Things is shown in Figure 1. The architecture model is composed of four parts: sensor nodes (devices), gateways, cloud servers, and users. They are interconnected through wired and/or wireless technologies. In general, a gateway connects and manages several corresponding sensing devices [6]. As most of sensing devices deployed in an IoT environment are physically accessible by adversaries, physical security of device hardware and information security for device communications have become serious concerns for the deployment of IoT infrastructure. In addition, heterogeneous hardware components and functionalities along with support complexity on multiple communication protocols among sensing devices and gateways have brought more security challenges to IoT environments. Among security issues [7,8] of IoT environments, such as data protection and access control, device authentication between a sensing device and a gateway is a fundamental and indispensable security feature.



**Figure 1.** The generic architecture of the Internet of Things.

Authentication is an indispensable security mechanism. The goal is to identify the legitimacy of an entity such as a device or a user. Based on previously published literature, user authentication is divided into two categories: static authentication and continuous authentication [9]. A general static user authentication process will be invoked at the beginning of a communicating session to authenticate the identity of a user, who is trying to log into a corresponding service server [9,10]. In general, a secret known, possessed or biologically inherited by the genuine user, such as password, PIN, smartcard, security token, face features, and fingerprint, will be used as the input of an authentication request [11,12]. Unfortunately, static authentication cannot defend against session hijacking attacks. In order to strengthen security, the concept of continuous authentication is developed. Continuous authentication can repeatedly authenticate the legitimacy of a user during the time of device usage. However, continuous authentication is not a substitute for static authentication [12]. In fact, it complements static authentication to enhance security strength. Existing approaches for continuous user authentication employ behavioral biometrics such as mouse and keystroke dynamics

to continuously check the authenticity of a user [10,13]. In [14], shared secrets were used to construct authentication tokens; after both communicating parties (the user and the server) agree on the secret used in this authentication session, in a pre-defined time interval only the tokens and corresponding messages are required to be transmitted from the user to the server. The server can verify the received messages are sent from the genuine user based on the associated tokens. In previously published literature, continuous authentication mechanisms were generally applied to user-to-device authentication models [10,13,14].

In an IoT environment, how to accomplish device-to-device authentication has become a practical and fundamental issue. Most sensing devices have limited computing resources and storage capacity [14]. These devices cannot perform complex computations such as encryption and decryption operations unless they have been equipped with a sufficient amount of flash memory and enough computing power microcontrollers (e.g., constrained devices categorized as class 2 in [15,16]). Existing approaches have proposed lightweight protocols to authenticate the legitimacy of a peer device when a message needs to transmit to the peer device [8,17–21]. However, it is possible that a lot of instantaneous messages are transmitted in a short time period between a sensor node and a gateway in an IoT environment. Under such circumstances, adopting existing device-to-device authentication solutions may consume a lot of time in the authentication process in comparison with the time required for processing the received message. Therefore, the goal of this study is to design a lightweight device-to-device continuous authentication protocol to authenticate each data message exchanged between two devices within a pre-defined time period under IoT environment conditions.

The proposed lightweight continuous authentication protocol has the following features: (i) the protocol supports mutual authentication, i.e., both peer devices can authenticate each other before transmitting any messages; (ii) the protocol only uses lightweight computation operations, which include hash-based message authentication (HMAC) [22], hash function, and bitwise exclusive-or (XOR) operation, such that most sensing devices with limited computing resource have a good chance to adopt this protocol and install the corresponding protocol module; (iii) the proposed protocol contains two phases: the static authentication phase dynamically generates an agreed initial token for both communicating parties and the continuous authentication phase transmits authenticated initial token along with sensed data from the sensor device to the gateway. The protocol adopts token technique to support continuous authentication in which the session token secretly contains the dynamic (or time-dependent) feature of the sensing device, i.e., the remaining battery capacity of the sensing device in the proposed scheme. In addition, the time-bounded concept of valid authentication time period is proposed to enhance the security robustness of authentication between IoT devices. Security analysis and performance analysis are conducted to evaluate security strength and time consumption of the proposed protocol. The proposed protocol can also be extended in two aspects: adopting the protocol implementation option of gateway-initialized request and adding the feature of identity anonymity onto sensing devices.

The rest of this paper is organized as follows: related literature surveys on IoT authentication and continuous authentication are addressed in Section 2. In Section 3, the proposed lightweight authentication protocol is depicted. In Section 4, security and performance analyses for the proposed protocol are presented. Section 5 presents possible extensions of the proposed protocol. The final conclusions are addressed in Section 6.

## 2. Related Work

As IoT environments are open or semi-open to their potential users in general, adversaries can easily access devices deployed within an IoT environment. Therefore, IoT environments are vulnerable to various security threats. In consequence, authentication mechanisms should be implemented to provide secure communication between devices. In the section, related literature on traditional authentication and continuous authentication for IoT environments are discussed.

### 2.1. IoT Authentication

In this subsection, we categorize existing device-to-device authentication protocols for IoT environments into three groups: certification-based authentication, encryption-based authentication, and non-encryption-based authentication.

- *Certification-based Authentication:* The Datagram Transport Layer Security (DTLS) [23] protocol is an existing standard. In 2013, Kothmayr et al. [24] proposed a two-way authentication security scheme for IoT based on DTLS, which used RSA-based asymmetric encryption and X.509 certification. However, this scheme needs eight handshakes to establish a session. Hence, in order to implement this scheme, higher consumption cost and more storage space are required from resource-constrained sensing devices. In 2014, Porambage et al. [21] proposed an authentication protocol by using implicit certificate in distributed IoT environments. Since Elliptic Curve Cryptography (ECC) consumes less computing resources relative to RSA, the protocol in [21] employs the Elliptic Curve Qu-Vanstone (ECQV) implicit certificate scheme and the Elliptic Curve Diffie-Hellman (ECDH) key exchange scheme. The protocol uses implicit certificates to accomplish end-to-end authentication in distributed IoT environments. The protocol contains two authentication phases, which are the registration phase and the authentication phase. Although the proposed scheme adopts ECC to reduce computation overhead for sensing devices, the protocol still requires some storage space in devices to store implicit certificates and the scheme also requires a Certificate Authority (CA).
- *Encryption-based Authentication:* In 2015, Shivraj et al. [8] proposed One Time Password (OTP) authentication for IoT infrastructures. The protocol adopts Identity Based Elliptic Curve Cryptography (IBE-ECC) to provide a lightweight end-to-end authentication between devices. The advantage of the protocol is that sensing devices do not need extra storage for storing the keys as the scheme uses OTP. However, if the devices need to communicate frequently, they must frequently request the central cloud to generate the OTP. In consequence, communicating devices may spend more time to establish a session. In 2015, Mahalle et al. [20] proposed a group authentication protocol for IoT environments. The protocol could effectively authenticate the devices in the same group. The proposed TCGA scheme uses Paillier Threshold Cryptography. The Paillier Cryptosystem possesses special properties such as homomorphic addition, indistinguishability, and self-binding [25,26]. The TCGA scheme establishes a session key for each group authentication to achieve efficient authentication among group members. The main concern for this scheme is that if a new device member joins the group, the keys for group members have to be regenerated and distributed to all members again. If the targeted IoT environment needs to frequently change device members in their group, it may cause additional authentication overhead for the devices. In 2015, Khemissa and Tandjaoui [18] proposed a lightweight authentication for IoT environments without using complex cryptographic operations. The protocol employed hash-based message authentication code (HMAC) [22] operations and nonce to establish mutual authentication. Advanced Encryption Standard (AES) [27] encryption mechanism was used to generate a session key. Hence, the scheme requires sensing devices to possess the ability to perform symmetric cryptographic operations. In 2016, Khemissa and Tandjaoui [19] extended their work in [18] to support remote users. The protocol could achieve mutual authentication between a sensor node and a remote user. A user could use his/her mobile device to manage heterogeneous sensing resources. In 2016, Kumar et al. [28] presented a lightweight authentication-based session key establishment protocol for smart home. The protocol requires a security service provider, which is a trusted server. The security service provider assigns important parameters, generates tokens and distributes tokens to communicating devices in a smart home environment. The devices use authenticated token to establish a session key and achieve mutual authentication.



- *Non-encryption-based Authentication:* In this category, proposed approaches do not use any certification technique or any encryption operation. In 2015, Gope et al. [29] proposed an untraceable authentication protocol in distributed IoT architecture. The scheme only uses hash functions and bitwise exclusive-or operations to construct a lightweight authentication mechanism. In addition, the scheme uses sequence numbers and random numbers to generate a one-time alias identity. The proposed scheme not only ensures the legality of a sensor node but also support identity anonymity and untraceability. In 2015, Kawamoto [30] presented a location-based authentication scheme in IoT environments. The protocol utilizes ambient information of devices for authentication. The scheme has to continuously collect ambient information from IoT devices.

## 2.2. Continuous Authentication

In this subsection, we review related literature on continuous authentication. We classify continuous authentication protocols into two categories: user-to-device models and device-to-device models:

- *User-to-Device Models:* Several schemes for continuous user authentication have been proposed in recent years [10,31–35]. The goal of these proposed solutions are to help devices to constantly authenticate the current user to prevent impersonated or illegal users using devices. The communication model of these schemes is user-to-device and most schemes utilize behavioral biometrics to construct their continuous authentication process. In 2010, Shimshon et al. [31] presented a continuous authentication mechanism which repeatedly verifies the identity of current device user based on keystroke dynamics. The proposed scheme collects multiple keystrokes from the genuine user to create corresponding feature vectors and use these vectors as the reference base. Once a genuine user gets authenticated to use the device with continuous authentication module, within pre-defined time period the module will repeatedly collect newly generated keystrokes, generate corresponding feature vectors and compare them with the reference base in order to validate the current user is indeed the one authenticated. In 2012, Shen et al. [32] proposed a continuous authentication protocol based on dynamic patterns of mouse usage by a genuine user. There are other approaches adopting multi-behavioral biometrics to construct continuous authentication mechanism. In 2014, Bailey et al. [33] proposed a continuous user authentication scheme using the combined patterns of keyboard, mouse, and Graphical User Interface (GUI) interactions generated from a user as the reference base of a specific genuine user to achieve higher authentication accuracy. In 2015, Buduru and Yau [10] introduced a continuous user authentication scheme based on the patterns of user finger gestures on the touch screen of a targeted device. Modified Markov Decision Process (MDP) models for different usage contexts are adopted by the scheme of Buduru and Yau. In 2010, Niinuma et al. [34] adopted soft biometric features including facial skin and clothing color to construct their continuous user authentication mechanism. In 2012, Mock et al. [35] proposed their continuous user authentication scheme based on a user iris recognition mechanism. This scheme could also add user password option to establish a multi-factor user authentication solution. In 2017, Peng et al. [36] introduced a continuous authentication mechanism for users who wear smart glasses to protect user privacy. This mechanism utilizes finger touch gestures and voice commands to construct their behavioral biometrics. In 2017, Zhou et al. [37] proposed a transparent authentication scheme to continuously authenticate targeted user through authentication token, which contains the brainwave patterns of the targeted user.
- *Device-to-Device Models:* To the best of our knowledge, there are no studies on device-to-device continuous authentication in IoT environments. In 2015, Bamasag and Youcef-Toumi [14] proposed a lightweight continuous user authentication for IoT environments. Their work identified the need of continuous authentication in IoT environments. As sensing devices in particular scenarios, such as personal health monitoring and industrial control systems [38], need to frequently transmit sensed data to gateways in a short period of time, a continuous authentication mechanism could accomplish faster authentication. In the proposed scheme,

secret shares are used to construct authentication tokens and only the tokens and corresponding messages are required to be transmitted from the user to the server in a pre-defined time interval for continuous authentication. The server can verify the received messages are sent from the genuine user based on the associated tokens. Even though the proposed protocol in [14] is under user-to-device model, it inspires us in many aspects to design a new lightweight device-to-device continuous authentication protocol.

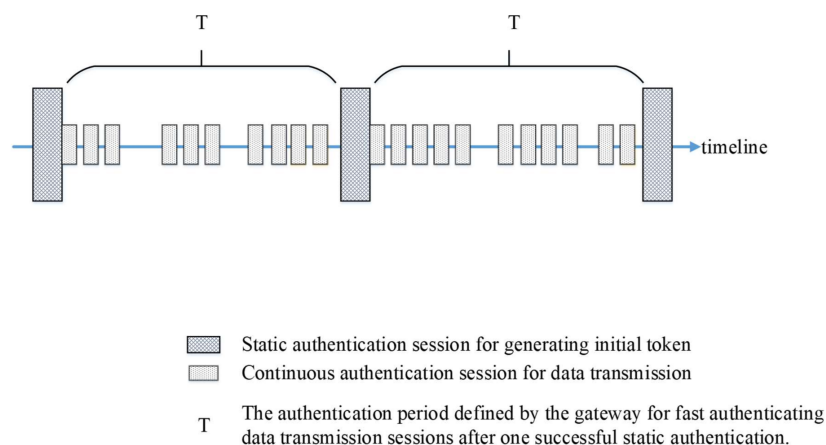
### 3. The Proposed Scheme

In this section, we introduce the proposed authentication protocol. We will describe our design concept, assumptions, and notations of our proposed protocol. The proposed protocol consists of three phases: initialization phase, static authentication phase, and continuous authentication phase.

#### 3.1. Design Concept

In some IoT environments such as factory monitoring and smart inpatient systems, sensor nodes frequently transmit a large number of sensed data to the gateway in a short time period. Since the time interval of each data transmission session is very short, the gateway must frequently authenticate the communicating devices (sensor nodes) in the beginning of each data transmission session. In order to fast ensure authenticity of devices for each received data in a valid session by a gateway, we adopt the design of continuous authentication. Continuous authentication can save authentication time for each data transmission session. The proposed protocol utilizes the value of remaining battery capacity as a dynamic factor to authenticate the targeted sensing device.

The proposed authentication protocol contains two authentication phases, which are the static authentication phase and the continuous authentication phase. In each phase, we have developed a corresponding authentication scheme. Static authentication scheme is similar to general or traditional authentication approaches and it is applied to authenticate devices in the beginning of an authentication period  $T$ . Continuous authentication scheme is applied to each sensed data transmission during the current authentication period  $T$ . To further clarify the proposed mechanism, Figure 2 shows the protocol framework through timeline, in which dot blocks indicate static authentication sessions and reticular blocks indicate continuous authentication sessions. If a sensor node transmits sensed data to a gateway, the sensor node and the gateway mutually authenticate each other during the static authentication phase. After passing the static authentication phase, the continuous authentication scheme is applied to each sensed data transmission from the sensor node to the gateway through the current authentication period  $T$ . There will be some time intervals in which no data transmission occurs during an authentication period  $T$ .



**Figure 2.** The proposed authentication protocol framework through timeline.

Within the pre-defined authentication period  $T$ , the static authentication process is first invoked for communicating devices to set up an authenticated token, which will be used for each continuous authentication session. Then, during the authentication period  $T$ , the gateway can quickly verify the legality of the sensor node when a new message or data needs to be transmitted between both parties. With the use of an authenticated token, the continuous authentication scheme spends less time for computation than the static authentication scheme. The proposed protocol does not utilize any cryptographic operation; therefore, it is a lightweight authentication protocol.

Next, we introduce two general scenarios in IoT environments to show the demand of lightweight device-to-device continuous authentication. The first scenario is factory monitoring as shown in Figure 3. For a factory monitoring system to be well functioned, many different types of sensing devices and intermediary gateways need to be deployed in a factory building. Each gateway can manage a number of sensing devices. Sensor nodes (devices) can obtain various sensed data such as machinery voltage, equipment vibration, machine temperature and pressure, and environmental information like humidity and temperature. The sensed data are collected by sensing devices and later transmitted to a gateway. After the gateway has received the data, it transmits sensed data to a cloud server or a local backend server. The sensed data are used to monitor the real-time production status of the factory. If there are abnormal situations occurred, corresponding safety alarms and warning messages will be generated immediately by the monitoring system. Since the system needs to gather real-time sensed data constantly, the sensing devices have to transmit their sensed data to intermediary gateways frequently in a short time period.

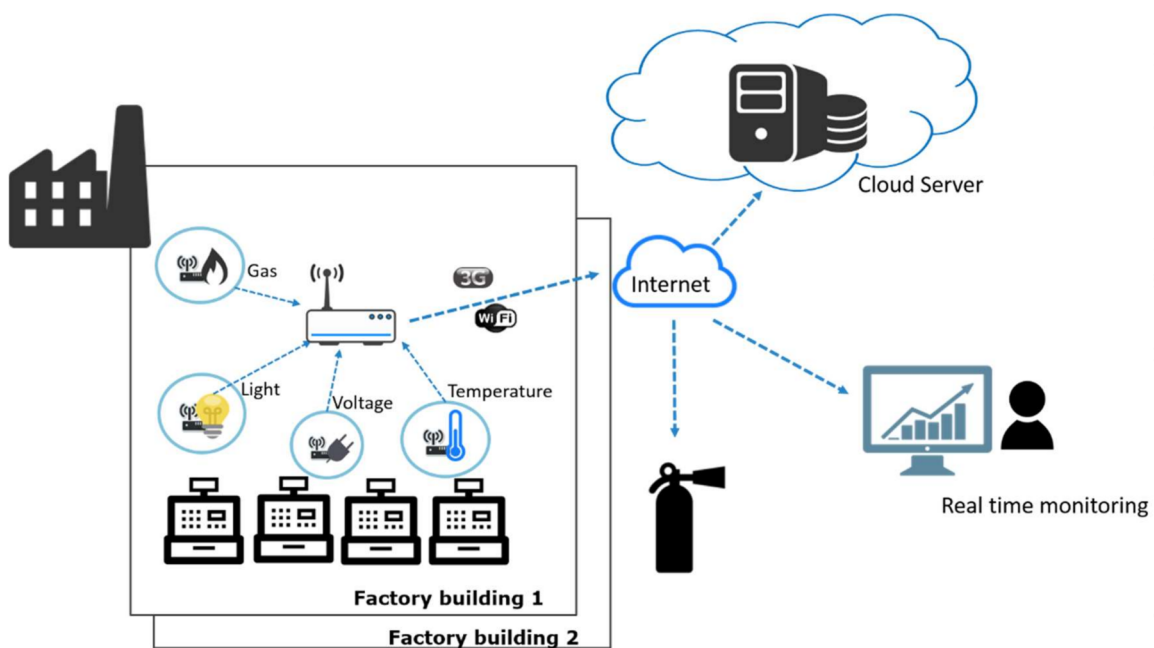


Figure 3. The possible IoT architecture for factory monitoring scenario.

The other scenario is the smart inpatient system as shown in Figure 4. To implement a smart inpatient system, the hospital needs to install and deploy various sensing devices in a ward or even asks the patients to wear some body sensors. These sensor nodes can acquire sensed data such as the patient's temperature, ECG, and blood pressure, and other environmental information. Similar to the first scenario, all sensed data will be collected to a cloud server or a local server. Doctors and nurses can watch over real-time body status of all inpatients and the environmental information of each ward through the system dashboard. In this scenario, the sensed data also needs to be transmitted periodically to the backend server in a short time period.

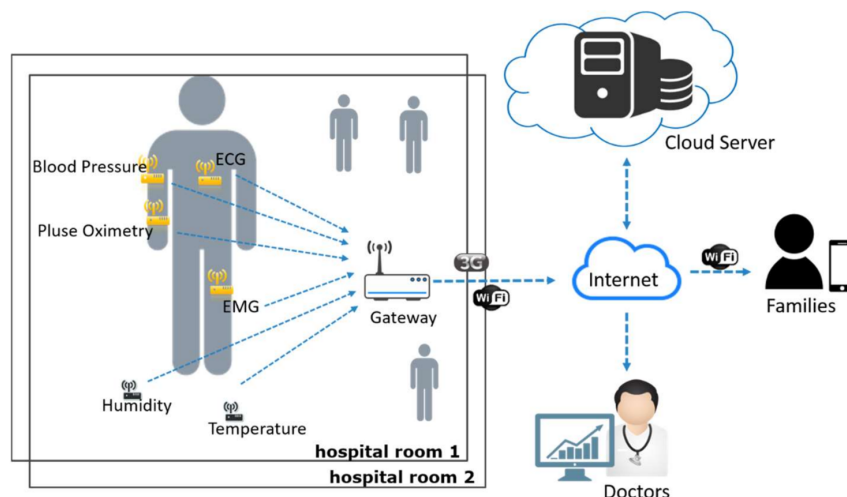


Figure 4. The possible IoT architecture for smart inpatient system scenario.

### 3.2. Assumptions

We list our assumptions for the proposed protocol as follows:

1. Sensor nodes are resource-limited devices powered by one or more batteries, which have low computational capability and storage space. Each sensor node is able to perform hash operation and has a random number generator to generate random numbers.
2. Gateways are resource-unlimited devices, which have sufficient computational capability to perform hash operation and generate random numbers, and storage space to store temporary values and pre-defined data tables.
3. Multiple sensor nodes can be managed by one gateway. Each sensor node and the corresponding gateway share one distinct secret value which is set in the initialization phase of the proposed authentication protocol.
4. The sensor node cannot precisely digitize and display its remaining energy (or battery) capacity on its display panel (if it has one).

### 3.3. Notations

The notations used are defined in Table 1.

Table 1. The notations and their definitions.

Notation	Definition
$SN$	A sensor node
$GW$	A gateway
$ID_{SN}$	The identity of a sensor node $SN$
$ID_{GW}$	The identity of a gateway $GW$
$AID_{SN}$	The anonymous identity of a sensor node $SN$
$T$	The authentication period defined by the gateway for fast authenticating data transmission sessions after one successful static authentication. The time unit is by minute
$t_s, t_c$	The timestamps
$H(\cdot)$	A one-way hash function
$SK_{SN}$	The secret value of a sensor node $SN$
$r_1, r_2, v$	Random numbers generated by a sensor node $SN$
$n_1, n_2, w$	Random numbers generated by a gateway $GW$
$HMAC_J(\cdot)$	Hash-based message authentication code function associated with the secret key $J$
$\parallel$	A concatenation operation
$\oplus$	A bitwise exclusive-or operation

Table 1. Cont.

Notation	Definition
$sd$	Sensed data from a sensor node $SN$
$ms$	The masked value of the sensed data from a sensor node $SN$
$rb$	The current energy capacity of sensor battery
$er$	The record of remaining energy capacity of sensor battery after last session
$mb$	The masked value of battery energy capacity
$TK_{SN}^I$	The initial token generated by a sensor node and the communicating gateway
$TK_{SN}^D$	The dynamic token generated by a sensor node
$EBC_{SN}$	The estimated daily average battery consumption value for a sensor node
$BCT_{SN}$	The estimated remaining battery capacity threshold for a sensor node to transmit data during a continuous authentication period $T$
$ACK$	Acknowledge message
$m, X, Y, Y_1, M_0, M_1, M_2, M_3, M_4, M_5$	Intermediate variables

### 3.4. Battery Consumption

- *Estimated Daily Average Battery Consumption ( $EBC_{SN}$ ):* In order to calculate the estimated daily average battery consumption value for a sensor node, we propose a daily battery consumption equation based on the battery life time and battery capacity of a sensor node:

$$EBC_{SN} = \frac{BC}{BL} \quad (1)$$

In Equation (1),  $EBC_{SN}$  indicates estimated daily average battery consumption value for a sensor node in the unit of milliampere-hour (mAh) per day.  $BC$  is the fully charged battery capacity of a sensor node in the unit of mAh.  $BL$  is the battery life time of a sensor node in the unit of day. We use battery life time and battery capacity (mAh) of a sensor node to calculate  $EBC_{SN}$  whose measurement unit is by mAh/day.

- *Estimated Remaining Battery Capacity Threshold ( $BCT_{SN}$ ):* In order to check the remaining battery capacity within a reasonable value for a sensor node during an authentication period  $T$ , we design an equation to compute the estimated remaining battery capacity threshold as shown in Equation (2):

$$BCT_{SN} = \left[ rb - \left( \frac{EBC_{SN}}{24 \times 60} \times T \right) \right] \times s \quad (2)$$

In Equation (2),  $BCT_{SN}$  indicates the estimated remaining battery capacity threshold of a sensor node and the measurement unit of  $BCT_{SN}$  is mAh.  $EBC_{SN}$  indicates the estimated daily average battery consumption value for a sensor node in the unit of mAh per day.  $rb$  is the current energy capacity of a sensor battery.  $\left( \frac{EBC_{SN}}{24 \times 60} \right)$  indicates the estimated battery consumption value per minute for the sensor node.  $\left( \frac{EBC_{SN}}{24 \times 60} \times T \right)$  indicates the estimated battery consumption value per authentication period  $T$  for the sensor node. For simplicity, we assume sensor battery consumption is a linear relationship in association with the running time of a sensor node. Symbol  $s$  indicates an estimation coefficient to accommodate possible deviation on the calculated threshold value  $\left[ rb - \left( \frac{EBC_{SN}}{24 \times 60} \times T \right) \right]$ . The calculated threshold value is multiplied by the coefficient  $s$  to form the final value of  $BCT_{SN}$ . In general, the battery consumption model along with the value of the coefficient  $s$  of a sensor device could be evaluated and revealed by the corresponding sensor vendor, where  $0 < s < 1$ .

### 3.5. The Proposed Authentication Protocol

This subsection introduces the details of our proposed protocol, which consists of three phases: initialization phase, static authentication phase, and continuous authentication phase.



### 3.5.1. Initialization Phase

In the Initialization phase, important parameters have to be set up for both sensor nodes and gateways. First, the sensor node submits its identity  $ID_{SN}$  and related battery information which includes battery life time and battery capacity to the gateway via a secure channel. There are many ways to deliver required secret values to both sensors and gateways; for example, sensor vendors can pre-install secret value into sensors during production phase and a gateway can acquire required secret values from a trusted third party server (a broker website or the Web service of targeted sensor vendor). If sensors are class 2 type devices as defined in [15], it is possible for a gateway to establish a secure channel with these sensors through encryption operations.

After the gateway receives the request from a sensor node, the gateway generates a secret value  $SK_{SN}$ . Subsequently, the gateway also computes the estimated daily average battery consumption value  $EBC_{SN}$  for a sensor node  $SN$ . Then, the gateway sets an authentication period which is defined by the gateway for fast authenticating data transmission sessions after one successful static authentication. Next, the gateway securely sends the secret value  $SK_{SN}$  back to the sensor node via a secure channel. As soon as the sensor node obtains the secret value  $SK_{SN}$  from the gateway, it stores the secret value  $SK_{SN}$  in its secure storage. Next, the gateway also records the sensor node's secret value  $SK_{SN}$ , authentication period  $T$ , and estimated daily average battery consumption value  $EBC_{SN}$  for a sensor node  $SN$  into its mapping table or database. Therefore, the gateway stores each managed sensor node's important information in the binding table whose field contains the identity  $ID_{SN}$ , the corresponding secret value  $SK_{SN}$ , authentication period  $T$ , and estimated daily average battery consumption value  $EBC_{SN}$  for the sensor node  $SN$ .

### 3.5.2. Static Authentication

In the static authentication phase, a sensor node and a gateway mutually authenticate each other within static authentication. Simultaneously, both sides negotiate an initial token  $TK_{SN}^I$  to be used for continuous authentication during the authentication period  $T$ . The gateway calculates estimated remaining battery capacity threshold  $BCT_{SN}$  which is used to check whether the remaining battery capacity value of a sensor node is reasonable when the gateway receives data sent from the sensor node in a data transmission session.

The static authentication phase of our proposed protocol is according to the following steps shown in Figure 5:

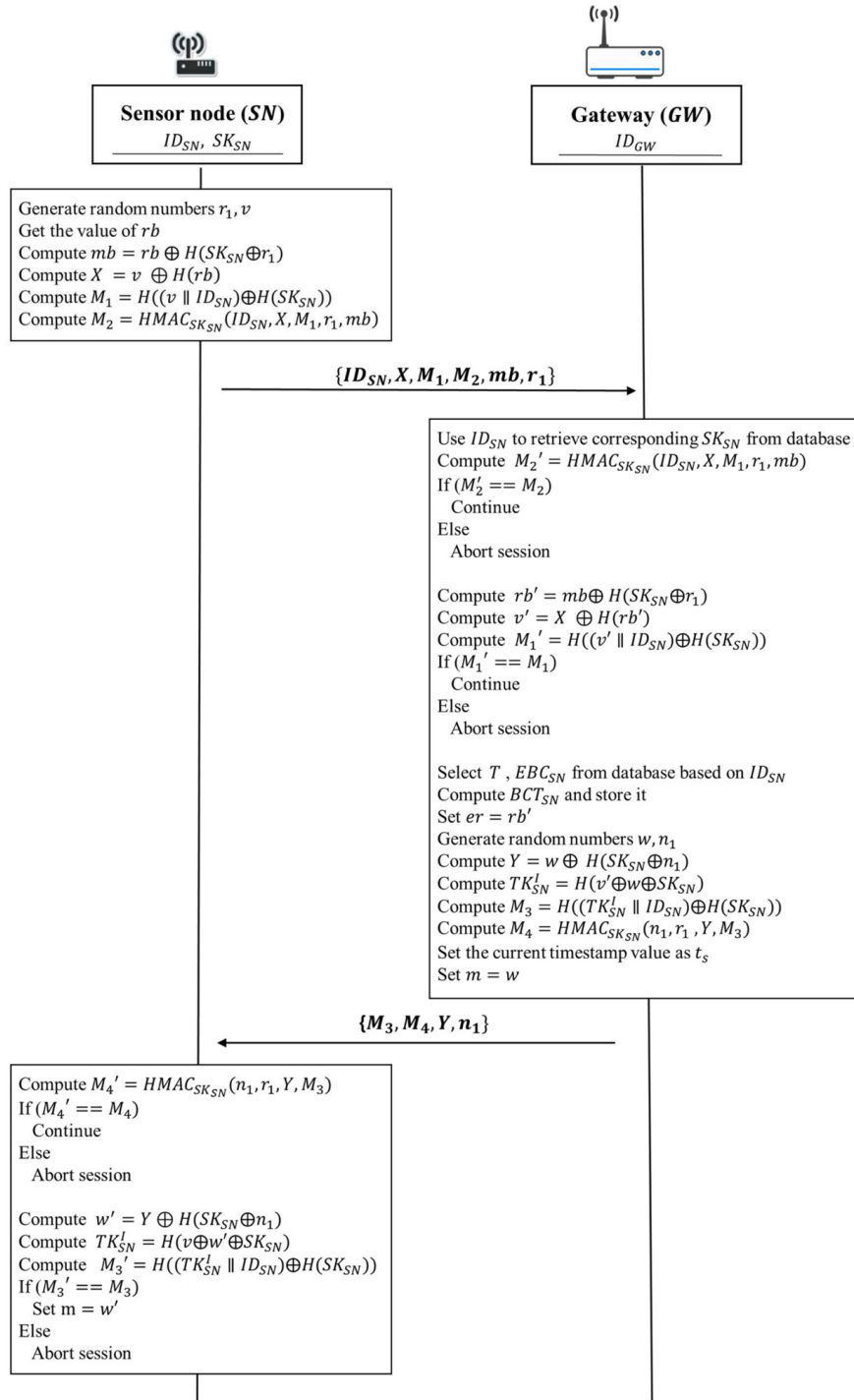
1. Sensor node  $\rightarrow$  Gateway:  $ID_{SN}, X, M_1, M_2, mb, r_1$

A sensor node generates random numbers  $r_1$  and  $v$ . Then the sensor node gets the value of the current energy capacity of sensor battery  $rb$  and gets its secret value  $SK_{SN}$  from its secure storage. Next, it takes the current energy capacity of sensor battery  $rb$  to compute  $mb = rb \oplus H(SK_{SN} \oplus r_1)$ . After that, the sensor node computes  $X = v \oplus H(rb)$  and  $M_1 = H((v \parallel ID_{SN}) \oplus H(SK_{SN}))$ , respectively. Subsequently, the sensor node uses the secret value  $SK_{SN}$  to compute  $M_2 = HMAC_{SK_{SN}}(ID_{SN}, X, M_1, r_1, mb)$  in order to guarantee that the message will not be modified during data transmission. Finally, the sensor node sends  $ID_{SN}, X, M_1, M_2, mb$ , and  $r_1$  to the gateway.

2. Gateway  $\rightarrow$  Sensor node:  $M_3, M_4, Y, n_1$

Upon receiving  $ID_{SN}, X, M_1, M_2, mb$ , and  $r_1$  from the sensor node, the gateway uses the identity of a sensor node  $ID_{SN}$  to retrieve corresponding secret value  $SK_{SN}$  from its database. Then the gateway uses the secret value  $SK_{SN}$  to compute  $M'_2 = HMAC_{SK_{SN}}(ID_{SN}, X, M_1, r_1, mb)$ . After that, the gateway verifies whether the computed value  $M'_2$  is equivalent to the received value  $M_2$ . If  $M'_2$  and  $M_2$  are equivalent, it indicates that the obtained messages are not changed by any malicious attacker. Otherwise, the gateway will terminate the protocol. Next, the gateway computes  $rb' = mb \oplus H(SK_{SN} \oplus r_1)$  and uses  $rb'$  to compute  $v' = X \oplus H(rb')$ . After computing  $rb'$  and  $v'$ , the gateway takes the random number  $v'$  to compute  $M'_1 = H((v' \parallel ID_{SN}) \oplus H(SK_{SN}))$ .

Then, the gateway verifies whether the computed value  $M_1'$  is equivalent to the received value  $M_1$ . If  $M_1'$  and  $M_1$  are equivalent, it indicates that the obtained values of  $v'$  and  $rb'$  are correct. Otherwise, the gateway will also terminate the protocol.



**Figure 5.** The static authentication phase of the proposed protocol.

After the above verification tasks have been completed, the gateway retrieves the corresponding authentication period  $T$  and the estimated daily average battery consumption value  $EBC_{SN}$  from its database associated with  $ID_{SN}$ . Subsequently, the gateway uses the authentication period  $T$  and the estimated daily average battery consumption value  $EBC_{SN}$  to compute the value of the estimated

remaining battery capacity threshold  $BCT_{SN}$  for the sensor node. Then the gateway sets  $er = rb'$ . After that, the gateway generates random numbers  $w$  and  $n_1$ . Then, the gateway performs the two computations to compute  $Y = w \oplus H(SK_{SN} \oplus n_1)$ ,  $TK_{SN}^I = H(v' \oplus w \oplus SK_{SN})$ , respectively. Next, the gateway uses the initial token  $TK_{SN}^I$  to compute  $M_3 = H((TK_{SN}^I \parallel ID_{SN}) \oplus H(SK_{SN}))$ . Subsequently, the gateway uses the secret value  $SK_{SN}$  and the random number  $n_1$  to compute  $M_4 = HMAC_{SK_{SN}}(n_1, r_1, Y, M_3)$ . Then the gateway stores  $BCT_{SN}$  and  $TK_{SN}^I$  in its database. It also sets current timestamp value as  $t_s$  and sets  $m = w$ . Finally, the gateway sends  $M_3$ ,  $M_4$ ,  $Y$ , and  $n_1$  to the sensor node.

Upon receiving  $M_3$ ,  $M_4$ ,  $Y$ , and  $n_1$  from the gateway, the sensor node uses the secret value  $SK_{SN}$  to compute  $M'_4 = HMAC_{SK_{SN}}(n_1, r_1, Y, M_3)$ . After the computation, the gateway verifies whether the computed value  $M'_4$  is equivalent to the received value  $M_4$ . If  $M'_4$  and  $M_4$  are equivalent, it indicates that all the obtained messages are not changed. Otherwise, the sensor node will terminate the protocol. Then the sensor node computes  $w' = Y \oplus H(SK_{SN} \oplus n_1)$ . Next, the sensor node uses the obtained random number  $w'$  to compute  $TK_{SN}^I = H(v \oplus w' \oplus SK_{SN})$ . After that, the sensor node uses the computed initial token  $TK_{SN}^I$  to compute  $M'_3 = H((TK_{SN}^I \parallel ID_{SN}) \oplus H(SK_{SN}))$ . Then, the sensor node verifies whether the computed value  $M'_3$  is equivalent to the received value  $M_3$ . If  $M'_3$  and  $M_3$  are the same, it indicates that the computed initial token  $TK_{SN}^I$  is correct and authentication task is successful. The sensor node sets  $m = w'$  and stores the initial token  $TK_{SN}^I$  in the secure storage of the sensor node. Then, the sensor node can perform the continuous authentication for each data transmission during the current authentication period  $T$ . Otherwise, the sensor node will terminate the protocol.

### 3.5.3. Continuous Authentication

Continuous authentication is applied to sensed data transmission from the sensor node to the gateway after static authentication between them has been validated during the current authentication period  $T$ . Since continuous authentication happens after one successful static authentication, the sensor node has stored initial token  $TK_{SN}^I$ , and estimated remaining battery capacity threshold  $BCT_{SN}$  for the current authentication period  $T$ . After receiving data from a sensor node, the gateway performs a series of verification tasks to ensure the authenticity of a sensor node. First, the gateway checks whether the received message is generated in current authentication period  $T$ . Second, the gateway verifies the value  $M_5$  which indicates data integrity of the transmitted message and the remaining battery capacity  $rb$  is within a reasonable range. Finally, the gateway sends an acknowledgement  $ACK$  to the sensor node.

The continuous authentication phase of our proposed protocol is according to the following steps shown in Figure 6:

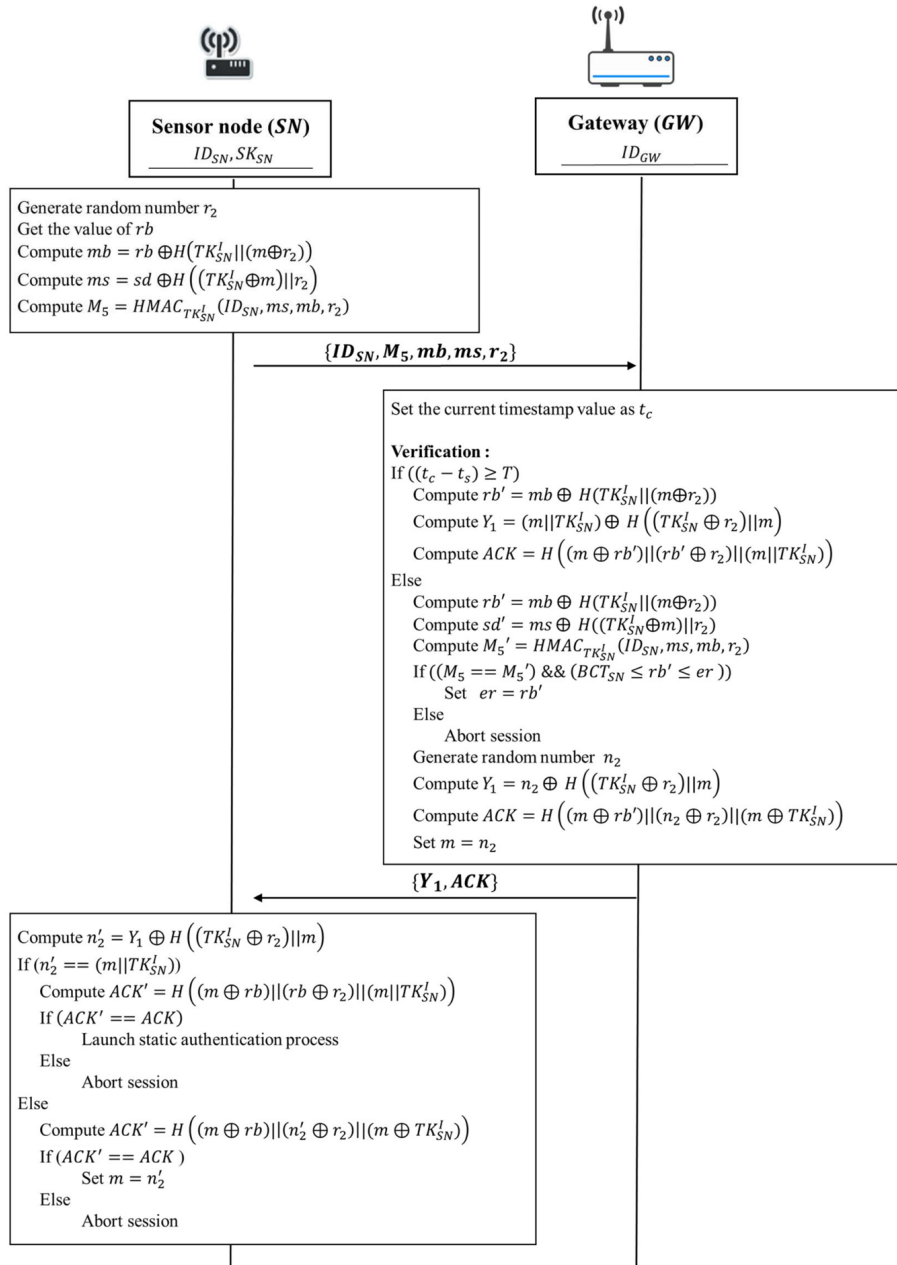
1. Sensor node  $\rightarrow$  Gateway:  $ID_{SN}, M_5, mb, ms, r_2$

Sensor node generates a random numbers  $r_2$  and gets the value of the current energy capacity of the sensor battery  $rb$ . Then, the sensor node gets the initial token  $TK_{SN}^I$  from its secure storage and takes the current energy capacity of sensor battery  $rb$  to compute  $mb = rb \oplus H(TK_{SN}^I \parallel (m \oplus r_2))$ . Next, the sensor node uses the random number  $m$  which had been generated at the static authentication phase to compute  $ms = sd \oplus H((TK_{SN}^I \oplus m) \parallel r_2)$  for masking the sensed data  $sd$ . After that, the sensor node computes  $M_5 = HMAC_{TK_{SN}^I}(ID_{SN}, ms, mb, r_2)$  in order to detect that the message has been modified during the data transmission. Next, the sensor node sends  $ID_{SN}$ ,  $M_5$ ,  $mb$ ,  $ms$ , and  $r_2$  to the gateway.

2. Gateway  $\rightarrow$  Sensor node:  $Y_1, ACK$

Upon receiving  $ID_{SN}$ ,  $M_5$ ,  $mb$ ,  $ms$ , and  $r_2$  from the sensor node, the gateway performs a series of verification tasks. First, the gateway sets the current timestamp value as  $t_c$ . Then, the gateway verifies whether the received message is generated in the current authentication period  $T$ .

If  $((t_c - t_s) \geq T)$ , it indicates that the timestamp  $t_c$  is out of range of the period time  $T$ . That is, the sensor has to launch the static authentication again for security issue. Therefore, the gateway compute  $rb' = mb \oplus H(TK_{SN}^I || (m \oplus r_2))$ ,  $Y_1 = (m || TK_{SN}^I) \oplus H((TK_{SN}^I \oplus r_2) || m)$  and  $ACK = H((m \oplus rb) || (rb \oplus r_2) || (m || TK_{SN}^I))$  for informing the sensor to relaunch the static authentication process. Then, the gateway sends an acknowledgement  $ACK$  and  $Y_1$  to the sensor node.



**Figure 6.** The continuous authentication phase of the proposed protocol.

After verifying timestamp, the gateway verifies data integrity of the message during the data transmission. Based on the received  $ID_{SN}$ , the gateway gets the corresponding initial token  $TK_{SN}^I$  from its database. The gateway uses the initial token  $TK_{SN}^I$  to compute  $b' = mb \oplus H(TK_{SN}^I || (m \oplus r_2))$ ,  $sd' = ms \oplus H((TK_{SN}^I \oplus m) || r_2)$  and  $M_5' = HMAC_{TK_{SN}^I}(ID_{SN}, ms, mb, r_2)$ , respectively. Then the gateway verifies whether the computed value  $M_5'$  is equivalent to the received value  $M_5$ . If  $M_5'$  and  $M_5$  are equivalent, it indicates that the obtained message is not modified during transmission.

In addition, the gateway also verifies whether the remaining battery capacity  $rb'$  is within a reasonable range. If  $(BCT_{SN} \leq rb' \leq er)$ , it indicates that the current energy capacity of sensor battery  $rb'$  is within a reasonable range and the validity of the sensor node is authenticated. Otherwise, the gateway aborts the session.

After the above verification tasks have been completed, sensor node is authenticated successfully and the gateway can assure the sensed data  $sd'$  which is transmitted by the sensor node is valid. The gateway sets  $er = rb'$ . Then the gateway generates a random number  $n_2$  to compute  $Y_1 = n_2 \oplus H((TK_{SN}^I \oplus r_2)||m)$  and  $ACK = H((m \oplus rb')||(n_2 \oplus r_2)||(m \oplus TK_{SN}^I))$ . After that, the gateway sets  $m = n_2$ . Finally, the gateway sends an acknowledgement  $ACK$  and  $Y_1$  to the sensor node.

Once the sensor node receives  $Y_1$  and  $ACK$ , the sensor node computes  $n'_2 = Y_1 \oplus H((TK_{SN}^I \oplus r_2)||m)$ . If the value of  $n'_2$  is equivalent to  $(m||TK_{SN}^I)$ , it means that this continuous authentication process is expired. That is, the sensor should relaunch the static authentication again. Before launching the static authentication, the sensor has to compute  $ACK' = H((m \oplus rb')|(rb \oplus r_2)|(m||TK_{SN}^I))$  to verify data integrity of the received message. If  $(ACK' == ACK)$ , it indicates that the message have not been modified during the data transmission. Then, the sensor relaunch the static authentication process. Otherwise, the sensor aborts the session.

On the other hand, if the value of  $n'_2$  is not equivalent to  $(m||TK_{SN}^I)$ , this continuous authentication should be successful. In order to verify data integrity of the received message, the sensor computes  $ACK' = H((m \oplus rb')|(n'_2 \oplus r_2)|(m \oplus TK_{SN}^I))$  to compare with the received value of  $ACK$ . If the value of  $ACK'$  is equivalent to  $ACK$ , this continuous authentication is successful. Finally, the sensor node sets  $m = n'_2$  and goes to the next continuous authentication for each data transmission. Otherwise, the protocol will be terminated. The sensor node must receive the acknowledgement  $ACK$  to indicate a graceful session ending.

## 4. Protocol Analysis

### 4.1. Security Analysis

In this section, a security analysis is conducted to evaluate the security robustness of the proposed protocol. There are six security properties supported by the protocol design: resistance to replay attack, resistance to impersonation attack, resistance to man-in-the-middle attack, data integrity, mutual authentication, and forward secrecy.

- *Resistance to Replay Attacks:* A malicious attacker may eavesdrop valid messages transmitted during an authentication session. Later on, the attacker replays some of these messages to impersonate a legitimate entity for establishing an authenticated session with the target peer. In our proposed protocol, if an attacker eavesdrops messages and performs a replay attack, the message receiver (the gateway or a sensor) can detect those messages are invalid. In our proposed protocol, the values  $M_2$ ,  $M_4$ ,  $M_5$ , and  $ACK$  containing transmitted messages are all constructed with fresh random numbers and each value will be transmitted along with its corresponding random number which is used as one of the variables to dynamically generate the value. These random numbers are freshly generated by both communicating parties during each authentication session. The message receiver will verify the validity of the received message by using the received random number to generate a tentative message and evaluating the equivalence between the received message and the tentative one. If both messages are equivalent, then the message receiver determines the received message is valid. As our protocol embeds random numbers into individual messages to keep freshness of the transmitting messages, our proposed protocol is able to resist any replay attack.
- *Resistance to Impersonation Attacks:* An impersonation attack indicates that a malicious attacker may try to masquerade as a valid sensor node. In the static authentication phase, if an attacker wants to masquerade as a sensor node, the attacker will need to forge the message  $\{ID_{SN}, X, M_1, M_2, mb, r_1\}$  sent to the gateway. If an attacker wants to forge the value  $M_2$ , the attacker needs to learn the



secret value  $SK_{SN}$ . The attacker may know the sensor identity  $ID_{SN}$  from the eavesdropped messages, but it is unable to learn the secret value  $SK_{SN}$ . Without knowing the secret value  $SK_{SN}$ , the attacker cannot compute a valid  $M_2$ . Therefore, the impersonation attack will fail. In the continuous authentication phase, if an attacker wants to masquerade as a valid sensor node, the attacker will need to forge the message  $\{ID_{SN}, M_5, mb, ms, r_2\}$  sent to the gateway. Therefore, the attacker needs to forge the value  $M_5$ . In consequence, the attacker needs to know the initial token  $TK_{SN}^I$ , the random numbers  $r_2$  and  $m$ , and the current energy capacity of sensor battery  $rb$ . The attacker may learn the random numbers  $r_2$  from eavesdropped messages but it cannot learn the initial token  $TK_{SN}^I$  from eavesdropped messages. Hence, the attacker cannot compute a valid initial token value  $TK_{SN}^I$  without knowing the value of the secret value  $SK_{SN}$ . In summary, the attacker cannot masquerade as a valid sensor node successfully. Hence, our protocol can resist the impersonation attack.

- Resistance to Man-in-the-middle Attacks:** A man-in-the-middle attack indicates that an active attacker secretly relays and manipulates the messages transmitted between two parties who believe they are directly communicating with each other. In our static authentication phase, if a malicious attacker wants to relay and manipulate transmitting messages, the attacker needs to learn the secret value  $SK_{SN}$  and the remaining energy capacity of sensor battery  $rb$ . Since the attacker cannot know the secret value  $SK_{SN}$  and the remaining energy capacity of sensor battery  $rb$  from previously eavesdropped messages, the attacker cannot learn the authentic data and manipulate messages successfully. In the continuous authentication phase, if the malicious attacker wants to relay and manipulate transmitting messages, the attacker needs to obtain the initial token  $TK_{SN}^I$ . As the initial token  $TK_{SN}^I$  is generated and securely sent in the static authentication phase, the attacker cannot learn the initial token  $TK_{SN}^I$ . The attacker can only eavesdrop the values of  $mb$  and  $ms$ , but it still cannot learn the authentic data  $rb$  and  $sd$  which are carefully hidden in  $mb$  and  $ms$ , accordingly. In consequence, the attacker cannot modify or manipulate transmitting messages without knowing the initial token  $TK_{SN}^I$ . Therefore, the proposed protocol can resist man-in-the-middle attacks.
- Data Integrity:** Data integrity indicates that a message receiver can ensure the message is not tampered with during transmission. Our protocol adopts an HMAC function to ensure data integrity. In our static authentication phase, if an attacker tries to tamper transmitting messages, the attacker needs to learn the secret value  $SK_{SN}$ . Since the attacker cannot learn the secret value  $SK_{SN}$  from eavesdropped messages, he cannot compute valid values  $M_2$  and  $M_4$  without knowing the secret value  $SK_{SN}$ . Hence, a malicious attacker cannot tamper transmitting messages successfully. In the continuous authentication phase, if an attacker tries to tamper transmitting messages, the attacker needs to learn the initial token  $TK_{SN}^I$ . As the attacker cannot learn the initial token  $TK_{SN}^I$  from eavesdropped messages, he cannot compute a valid value  $M_5$  without knowing the initial token  $TK_{SN}^I$ . Hence, our proposed protocol achieves the data integrity property.
- Mutual Authentication:** Mutual authentication indicates that two entities can authenticate each other. In the static authentication phase, the gateway authenticates the sensor node by verifying the value  $M_2$  with the shared secret value  $SK_{SN}$ . If the computed value  $M_2'$  is equivalent to the received value  $M_2$ , the gateway will be able to ensure the validity of the sensor node. Next, the sensor node also authenticates the gateway by verifying the value  $M_4$  with the shared secret value  $SK_{SN}$ . The value  $M_4$  embeds the random numbers  $r_1$  and  $n_1$ . If the computed value  $M_4'$  is equivalent to the received value  $M_4$ , the sensor node ensures the validity of the gateway. In the continuous authentication phase, both sensor and gateway can authenticate each other via initial token  $TK_{SN}^I$  and transmitted random numbers. First, the gateway authenticates the sensor node by verifying the value of  $M_5$  which is encrypted by the initial token  $TK_{SN}^I$ . If the value of  $M_5'$  is equivalent to  $M_5$ , the gateway ensures that the sensor node is valid. Next, the sensor node authenticates the gateway by verifying the value  $ACK$  which is composed of the

initial token  $TK_{SN}^I$  and random numbers  $n_2$  and  $r_2$ . Therefore, our proposed protocol supports mutual authentication between a sensor node and the gateway.

- *Forward Secrecy*: The aim of forward secrecy is to protect session keys generated in the past against compromises of session keys generated in the future. If the initial token  $TK_{SN}^I$  is learned by an attacker who wants to derive the initial token  $H(v \oplus w \oplus SK_{SN})$  used in the previous session, the attacker needs to know the previously generated random numbers  $v$  and  $w$ . The random numbers  $v$  and  $w$  were generated by the sensor node and the gateway in the past authentication session. Since the attacker cannot obtain previously generated random numbers  $v$  and  $w$  from the previous eavesdropped messages, therefore, the attacker cannot use the current initial token  $TK_{SN}^I$  to derive the previous initial token. Hence, our proposed protocol has the property of forward secrecy.

In order to evaluate the security strength of the proposed protocol, an automatic verification tool for security protocols called *Scyther* [39] is used. *Scyther* adopts formal analysis methodology and assumes perfect cryptography, in which it is assumed that all cryptographic functions are perfect. An adversary learns nothing from an encrypted message unless he knows the decryption key. *Scyther* has been used to evaluate many practical protocols [39]. *Scyther* can analyze a target protocol for secrecy and authentication in terms of aliveness, weak agreement, non-injective agreement, and non-injective synchronization [40,41]. A security protocol definition language (spdl) is invented and used for *Scyther* to describe the details of the protocol which will be verified. Due to the syntax limitation of *Scyther*, we write two spdl scripts to describe two different conditions of the continuous authentication phase of the proposed protocol:

- Condition (1): The sensor has to relaunch the static authentication phase when the gateway identifies the value of  $(t_c - t_s)$  is larger than or equal to the valid authentication time period  $T$ . In this case, the gateway will construct special values of  $Y_1$  and  $ACK$ . Once the sensor receives  $Y_1$  and  $ACK$  from the gateway, it can determine whether relaunching the static authentication phase is required by evaluating the extracted value of  $n'_2$  from the received  $Y_1$ .
- Condition (2): Normal continuous authentication phase is executed.

The spdl scripts for the static authentication phase and the continuous authentication phase of the proposed protocol are shown in Figures 7 and 8. Figures 9–11 show the execution results of our spdl scripts using *Scyther*. These results indicate no attacks are found within bounds, which were set as the maximum number of rounds. The term “no attacks within bounds” shown in Figures 9–11 indicates that *Scyther* did not find any attacks by reaching the bound. The verification results prove that the proposed protocol is secure.

#### 4.2. Performance Analysis

As mentioned in Section 2.2, we could not find other existing lightweight continuous authentication protocols to compare with the proposed protocol. Therefore, we planned to find a fast traditional user-to-device authentication protocol to compare with our protocol. In this subsection, we compare our proposed protocol with the protocol of Khemissa et al. [18] in terms of performance efficiency. The reason for selecting the protocol in [18] to compare with ours is that the protocol in [18] is a near lightweight authentication protocol. Therefore, its computation cost is already less than other existing authentication protocols we surveyed in Section 2.

Since the time consumption of executing a concatenation operation and bitwise exclusive-or operation is much less than other computing operations, we ignore time consumption generated by these two operations when calculating computation cost for an authentication protocol. As sensor nodes only equip limited computing resources and gateways usually have unlimited computing resources, we then focus on comparing the time consumption of the authentication process at the sensor node side between targeted protocols.

```

45 role S {
46
47   macro v = GetRandom;
48   fresh r2: Nonce;
49   macro rb = GetBatteryCap;
50   macro sd = GetMessagePayload;
51
52   macro mb = XOR(rb, SHA1(OR(tksNi, XOR(m, r2))));
53   macro ms = XOR(sd, SHA1(OR(XOR(tksNi, m), r2)));
54   macro m5 = SHA1PMAC (CONCAT(idSN, ms, mb, r2, tksNi));
55
56   var ack;
57   var y1;
58
59   send_1(S, G, idSN, m5, mb, ms, r2);
60   recv_2(G, S, y1, ack);
61   claim(S, Running, G, y1, ack);
62   macro n2p = XOR(y1, SHA1(OR(XOR(tksNi, r2), m)));
63   match(n2p, OR(m, tksNi));
64   claim_s1(S, Secret, rb);
65   claim_s2(S, Secret, skSN);
66   claim_s3(S, Alive);
67   claim_s4(S, Weakagree);
68   claim_s5(S, Commit, G, y1, ack);
69   claim_s6(S, Niagree);
70
71 };
72
73 role G {
74
75   var idSN;
76   var m5;
77   var mb;
78   var ms;
79   var r2: Nonce;
80
81   recv_1(S, G, idSN, m5, mb, ms, r2);
82   claim(G, Running, S, idSN, m5, mb, ms, r2);
83   macro rbp = XOR(mb, SHA1(OR(tksNi, XOR(m, r2))));
84   macro y1 = XOR(OR(m, tksNi), SHA1(OR(XOR(tksNi, r2), m)));
85   macro ack = SHA1(OR(XOR(m, rbp), XOR(rbp, r2), OR(m, tksNi)));
86   send_2(G, S, y1, ack);
87   claim_g1(G, Secret, rbp);
88   claim_g2(G, Secret, skSN);
89   claim_g3(G, Secret, tksNi);
90   claim_05(G, Alive);
91   claim_06(G, Weakagree);
92   claim_07(G, Commit, S, idSN, m5, mb, ms, r2);
93   claim_08(G, Niagree);
94
95 };

```

Figure 7. The spdl script for the static authentication phase of the proposed protocol.

<pre> 45 role S { 46 47   macro v = GetRandom; 48   fresh r2: Nonce; 49   macro rb = GetBatteryCap; 50   macro sd = GetMessagePayload; 51 52   macro mb = XOR(rb, SHA1(OR(tksNi, XOR(m, r2)))); 53   macro ms = XOR(sd, SHA1(OR(XOR(tksNi, m), r2))); 54   macro m5 = SHA1PMAC (CONCAT(idSN, ms, mb, r2, tksNi)); 55 56   var ack; 57   var y1; 58 59   send_1(S, G, idSN, m5, mb, ms, r2); 60   recv_2(G, S, y1, ack); 61   claim(S, Running, G, y1, ack); 62   macro n2p = XOR(y1, SHA1(OR(XOR(tksNi, r2), m))); 63   match(n2p, OR(m, tksNi)); 64   claim_s1(S, Secret, rb); 65   claim_s2(S, Secret, skSN); 66   claim_s3(S, Alive); 67   claim_s4(S, Weakagree); 68   claim_s5(S, Commit, G, y1, ack); 69   claim_s6(S, Niagree); 70 71 }; 72 73 role G { 74 75   var idSN; 76   var m5; 77   var mb; 78   var ms; 79   var r2: Nonce; 80 81   recv_1(S, G, idSN, m5, mb, ms, r2); 82   claim(G, Running, S, idSN, m5, mb, ms, r2); 83   macro rbp = XOR(mb, SHA1(OR(tksNi, XOR(m, r2)))); 84   macro y1 = XOR(OR(m, tksNi), SHA1(OR(XOR(tksNi, r2), m))); 85   macro ack = SHA1(OR(XOR(m, rbp), XOR(rbp, r2), OR(m, tksNi))); 86   send_2(G, S, y1, ack); 87   claim_g1(G, Secret, rbp); 88   claim_g2(G, Secret, skSN); 89   claim_g3(G, Secret, tksNi); 90   claim_05(G, Alive); 91   claim_06(G, Weakagree); 92   claim_07(G, Commit, S, idSN, m5, mb, ms, r2); 93   claim_08(G, Niagree); 94 95 }; </pre>	<pre> 42 role S { 43 44   macro v = GetRandom; 45   fresh r2: Nonce; 46   macro rb = GetBatteryCap; 47   macro sd = GetMessagePayload; 48 49   macro mb = XOR(rb, SHA1(OR(tksNi, XOR(m, r2)))); 50   macro ms = XOR(sd, SHA1(OR(XOR(tksNi, m), r2))); 51   macro m5 = SHA1PMAC (CONCAT(idSN, ms, mb, r2, tksNi)); 52 53   var ack; 54   var y1; 55 56   send_1(S, G, idSN, m5, mb, ms, r2); 57   recv_2(G, S, y1, ack); 58   claim(S, Running, G, y1, ack); 59 60   macro n2p = XOR(y1, SHA1(OR(XOR(tksNi, r2), m))); 61   macro ackp = SHA1(OR(XOR(m, rb), XOR(n2p, r2), OR(m, tksNi))); 62   match(ack, ackp); 63   claim_s1(S, Secret, y1); 64   claim_s2(S, Secret, ack); 65   claim_s3(S, Secret, tksNi); 66   claim_s4(S, Alive); 67   claim_s5(S, Weakagree); 68   claim_s6(S, Commit, G, y1, ack); 69   claim_s7(S, Niagree); 70 71 }; 72 73 role G { 74 75   var idSN; 76   var m5; 77   var mb; 78   var ms; 79   var r2: Nonce; 80 81   recv_1(S, G, idSN, m5, mb, ms, r2); 82   claim(G, Running, S, idSN, m5, mb, ms, r2); 83 84   macro rbp = XOR(mb, SHA1(OR(tksNi, XOR(m, r2)))); 85   macro sdp = XOR(ms, SHA1(OR(XOR(tksNi, m), r2))); 86   macro m5p = SHA1PMAC (CONCAT(idSN, ms, mb, r2, tksNi)); 87   match(m5, m5p); 88 89   fresh n2: Nonce; 90 91   macro y1 = XOR(n2, SHA1(OR(XOR(tksNi, r2), m))); 92   macro ack = SHA1(OR(XOR(m, rbp), XOR(n2, r2), OR(m, tksNi))); 93   send_2(G, S, y1, ack); 94   claim_g1(G, Secret, rbp); 95   claim_g3(G, Secret, tksNi); 96   claim_05(G, Alive); 97   claim_06(G, Weakagree); 98   claim_07(G, Commit, S, idSN, m5, mb, ms, r2); 99   claim_08(G, Niagree); 100 101 }; </pre>
---	---

(a)

(b)

Figure 8. (a) The spdl script for the continuous authentication phase in Condition (1); (b) The spdl script for the continuous authentication phase in Condition (2).

Claim				Status	Comments
Static	S	Static,s1	Secret GetBatteryCap	Ok	No attacks within bounds.
		Static,s2	Secret skSN	Ok	No attacks within bounds.
		Static,s3	Alive	Ok	No attacks within bounds.
		Static,s4	Weakagree	Ok	No attacks within bounds.
		Static,s5	Commit G,m3,m4,y,n1	Ok	No attacks within bounds.
		Static,s6	Niagree	Ok	No attacks within bounds.
G		Static,g1	Secret GetBatteryCap	Ok	No attacks within bounds.
		Static,g2	Secret skSN	Ok	No attacks within bounds.
		Static,g3	Secret SHA1(XOR(XOR(XOR(GetRandom,SHA1(GetBatteryC...	Ok	No attacks within bounds.
		Static,05	Alive	Ok	No attacks within bounds.
		Static,06	Weakagree	Ok	No attacks within bounds.
		Static,07	Commit S,idSN,XOR(GetRandom,SHA1(GetBatteryCap)),S...	Ok	No attacks within bounds.
		Static,08	Niagree	Ok	No attacks within bounds.
		Done.			

Figure 9. The security analysis result of the static authentication phase of the proposed protocol.

Claim				Status	Comments
ContinuousA	S	ContinuousA,s1	Secret GetBatteryCap	Ok	No attacks within bounds.
		ContinuousA,s2	Secret skSN	Ok	No attacks within bounds.
		ContinuousA,s3	Alive	Ok	No attacks within bounds.
		ContinuousA,s4	Weakagree	Ok	No attacks within bounds.
		ContinuousA,s5	Commit G,y1,ack	Ok	No attacks within bounds.
		ContinuousA,s6	Niagree	Ok	No attacks within bounds.
G		ContinuousA,g1	Secret XOR(XOR(GetBatteryCap,SHA1(OR(tkSNi,XOR(m,r...	Ok	No attacks within bounds.
		ContinuousA,g2	Secret skSN	Ok	No attacks within bounds.
		ContinuousA,g3	Secret tkSNI	Ok	No attacks within bounds.
		ContinuousA,05	Alive	Ok	No attacks within bounds.
		ContinuousA,06	Weakagree	Ok	No attacks within bounds.
		ContinuousA,07	Commit S,idSN,SHA1HMAC(CONCAT(idSN,XOR(GetMessageP...	Ok	No attacks within bounds.
		ContinuousA,08	Niagree	Ok	No attacks within bounds.
		Done.			

Figure 10. The security analysis result of the continuous authentication phase in Condition (1).

Claim	Status	Comments
ContinuousA S ContinuousA,s1 Secret y1	Ok	No attacks within bounds.
ContinuousA,s2 Secret ack	Ok	No attacks within bounds.
ContinuousA,s3 Secret tkSNi	Ok	No attacks within bounds.
ContinuousA,s4 Alive	Ok	No attacks within bounds.
ContinuousA,s5 Weakagree	Ok	No attacks within bounds.
ContinuousA,s6 Commit G,y1,ack	Ok	No attacks within bounds.
ContinuousA,s7 Niagree	Ok	No attacks within bounds.
G ContinuousA,g1 Secret XOR(XOR(GetBatteryCap,SHA1(OR(tkSNi,XOR(m,r...	Ok	No attacks within bounds.
ContinuousA,g3 Secret tkSNi	Ok	No attacks within bounds.
ContinuousA,05 Alive	Ok	No attacks within bounds.
ContinuousA,06 Weakagree	Ok	No attacks within bounds.
ContinuousA,07 Commit S,idSN,SHA1HMAC(CONCAT(idSN,XOR(GetMessageP...	Ok	No attacks within bounds.
ContinuousA,08 Niagree	Ok	No attacks within bounds.

**Figure 11.** The security analysis result of the continuous authentication phase in Condition (2).

We define the following notations to indicate time consumption for different computing operations:

1.  $T_{Hash}$ : The consumed time of executing a hash function
2.  $T_{AES}$ : The consumed time of executing an AES operation
3.  $T_{HMAC}$ : The consumed time of executing a HMAC operation
4.  $T_{Random}$ : The consumed time of generating a random number.

Table 2 shows the comparison result of computation cost between the protocol of Khemissa et al. [18] and our proposed protocol. In the static authentication phase, our proposed protocol requires  $4T_{Random} + 16T_{Hash} + 4T_{HMAC}$  while the protocol in [18] needs  $2T_{Random} + 2T_{Hash} + 4T_{HMAC} + 2T_{AES}$ . In our proposed protocol, the lengths of  $ID_{SN}$ , random numbers  $(r_1, r_2, n_1, n_2, v, w)$ , and  $SK_{sn}$  are all 128 bits.

**Table 2.** The comparison result on computation cost between the protocol of Khemissa et al. and the proposed protocol.

Phase	Khemissa et al. [15]	Our Protocol
Static authentication	$2T_{Random} + 2T_{Hash} + 4T_{HMAC} + 2T_{AES}$	$4T_{Random} + 16T_{Hash} + 4T_{HMAC}$
Continuous authentication	-	Condition (1): $2T_{Random} + 9T_{Hash} + 1T_{HMAC}$
		Condition (2): $2T_{Random} + 8T_{Hash} + 2T_{HMAC}$

Based on the work in [42], the time consumption of AES encryption operation  $T_{AES}$  is approximately 2.76 ms, the time consumption of hash operation  $T_{Hash}$  is approximately 1.5 ms and the time consumption of HMAC operation  $T_{HMAC}$  is approximately 3.54 ms. A pseudorandom number generation operation is approximately 0.65 ms as shown in [43]. Our protocol needs 12 extra hash function and two random number generation operations to complete static authentication in



comparison with the protocol in [18]. On the other hand, our protocol reduces two AES encryption operation when comparing with the protocol in [18]. The continuous authentication phase of the proposed protocol only requires half of the computation operations of the static authentication phase. Notice that the time consumption between Condition (1) and Condition (2) has a slight difference as one hash operation using in Condition (1) is substituted by one HMAC operation in Condition (2).

In conclusion, the computation cost of the static authentication phase of the proposed protocol is around 40.76 ms while the computation cost of the protocol in [18] only takes 23.98 ms. However, the computation cost of the continuous authentication phase of the proposed protocol is around 18.34~20.38 ms. Notice that the computation cost of the protocol in [18] only counts for the authenticated session key generation. There will be extra computation cost when both communicating parties start to encrypt their transmitted messages using the agreed session key. In contrast, our protocol only generates the initial token agreed by the sensor device and the gateway in the static authentication phase. Within the continuous authentication phase, sensed data along with the initial token and other temporary control values are transmitted from the sensor to the gateway. Therefore, our proposed protocol has better performance on continuous authentication in terms of computation cost. In summary, the proposed protocol is a very competitive protocol in terms of performance efficiency.

## 5. Discussion

In this section, two possible extensions of our proposed protocol are discussed: gateway initializing request and identity anonymity.

### 5.1. Gateway Initializing Request

In a general scenario, sensor nodes consecutively collect sensed data and send them to a gateway based on a scheduled time frame or the moment when the storage space of a sensor node is getting full. In special situations, the gateway may actively send a request to a targeted sensor node and ask for its sensed data. After receiving the request, the sensor node sends sensed data to the gateway immediately. By partially modifying our proposed protocol, our design can easily support the need for the gateway to initialize communication with a sensor node. The static authentication phase of the modified protocol to support gateway initializing request is shown in Figure 12.

In the static authentication phase, the gateway generates a random number  $n_1$  and computes  $M_0 = \text{HMAC}_{SK_{SN}}(ID_{SN} \parallel n_1)$  first. Then, the gateway sends  $n_1$  and  $M_0$  as a request to the targeted sensor node to ask for sensed data. After the sensor node receives  $n_1$  and  $M_0$ , the sensor node uses the identity of the sensor node  $ID_{SN}$  to compute  $M'_0 = \text{HMAC}_{SK_{SN}}(ID_{SN} \parallel n_1)$ . If the computed value  $M'_0$  and the received value  $M_0$  are equivalent, the sensor node ensures the authenticity of the gateway. The rest of authenticating steps in the modified protocol are identical to the steps in the original proposed protocol. If necessary, the continuous authentication phase will be executed by both peers for the sensor node to keep transmitting sensed data to the gateway. The continuous authentication phase of the modified protocol is identical to the one in the original proposed protocol.

### 5.2. Identity Anonymity

In some occasions, identity anonymity of a sensor node may be required. For the proposed protocol to achieve identity anonymity on sensor nodes, random number and hash functions are adopted to mask the actual identity of each sensor node and guarantee the uniqueness of each anonymous sensor identity  $AID_{SN}$ . A malicious attacker cannot derive the original identity of a sensor node  $ID_{SN}$  from its anonymous identity  $AID_{SN}$ . The gateway needs additional computation time to determine the identity of a sensor node  $ID_{SN}$  from a received  $AID_{SN}$ . The static authentication phase of our proposed protocol associated with the identity anonymity feature is shown in Figure 13 and the continuous authentication phase with the identity anonymity feature is shown in Figure 14.

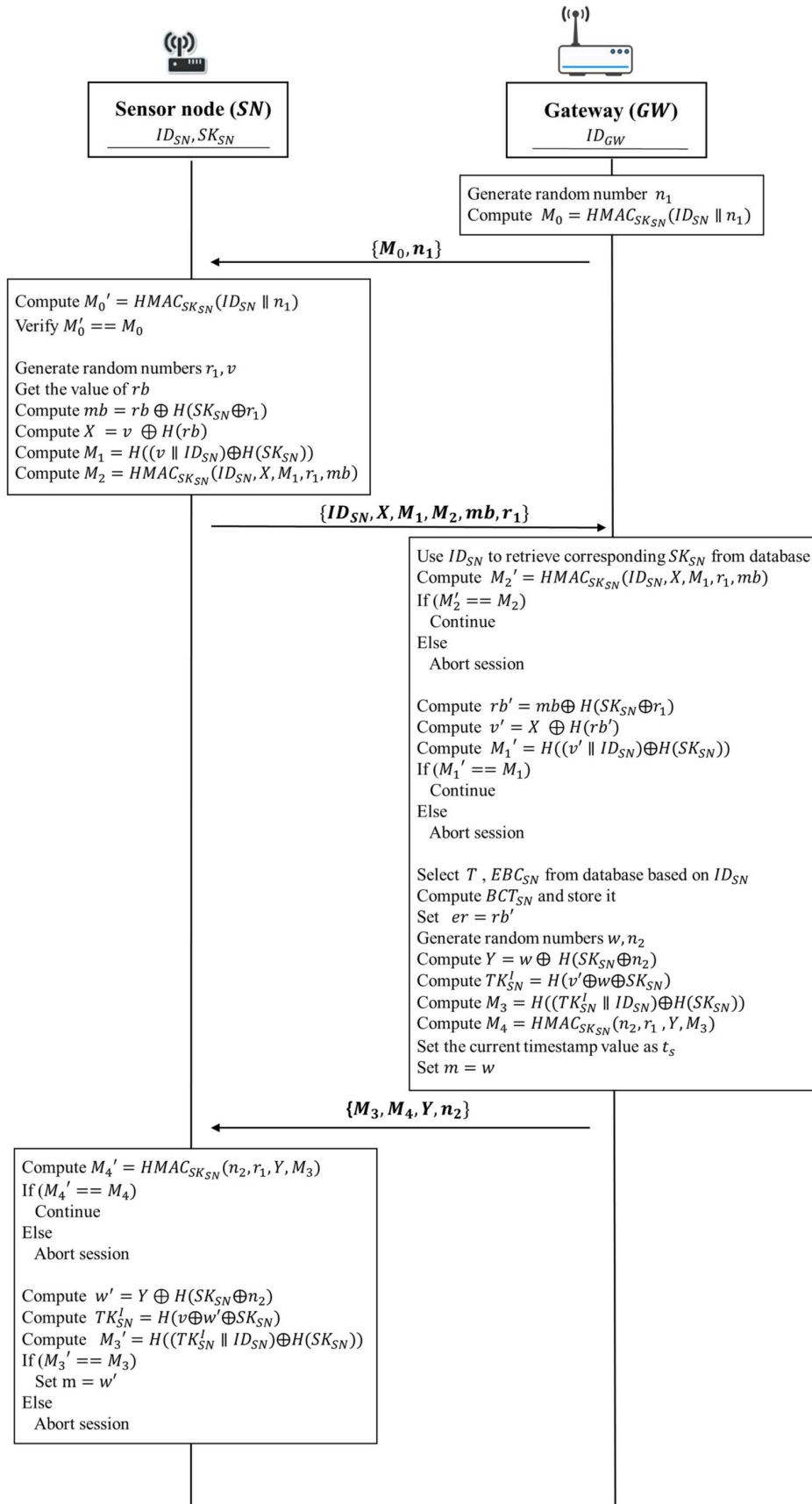


Figure 12. The static authentication phase for gateway initializing request.

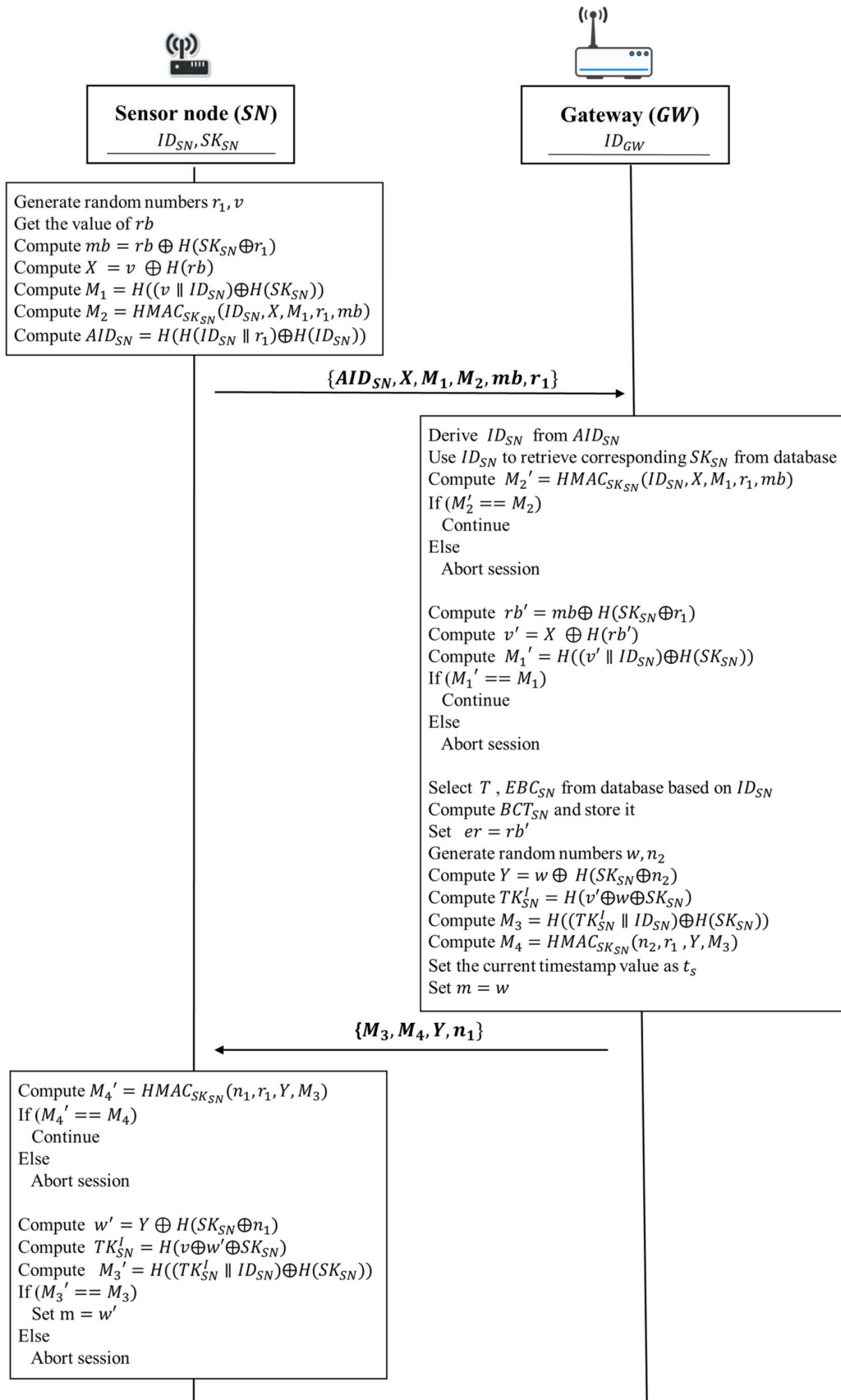
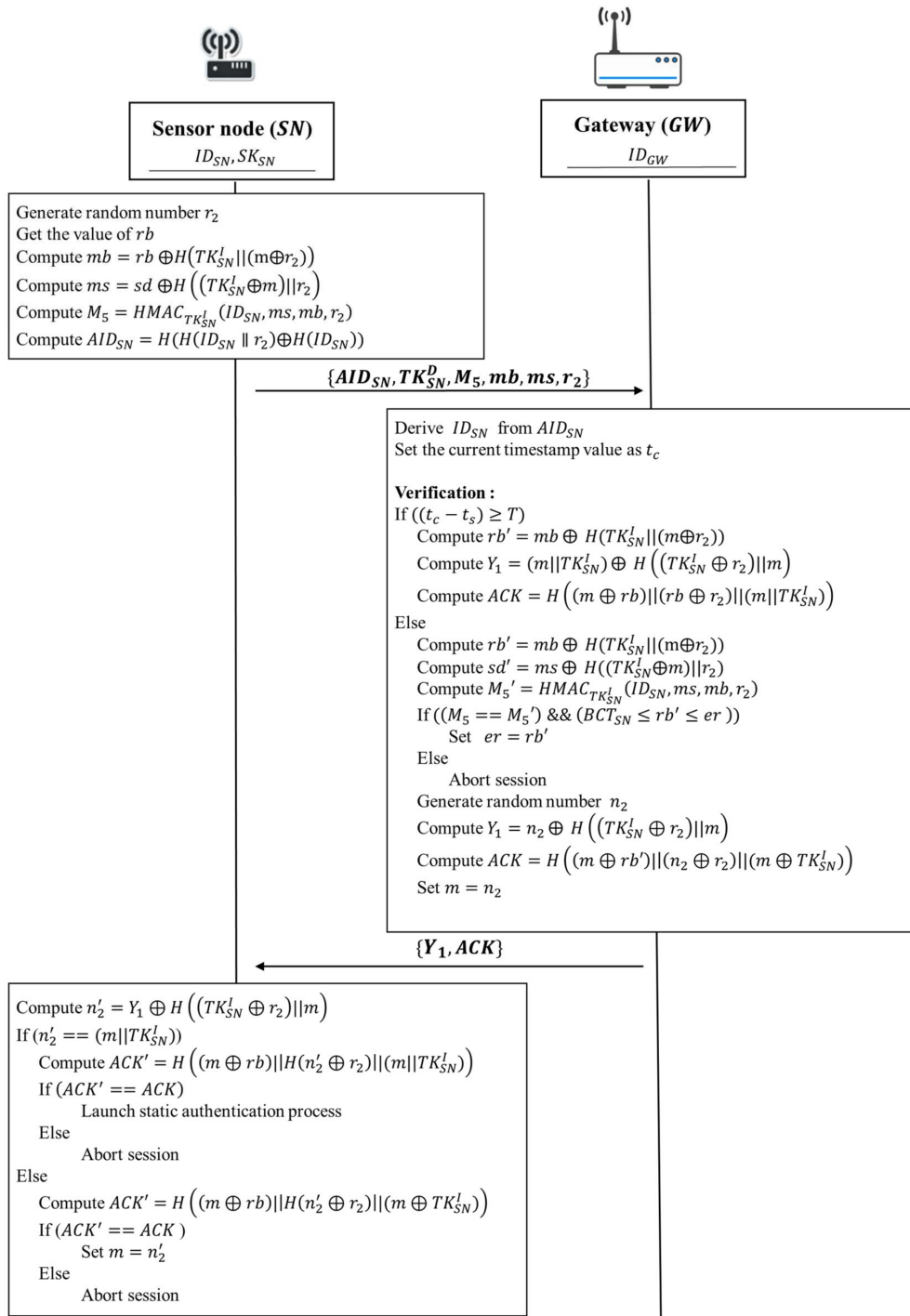


Figure 13. The static authentication phase with identity anonymity.



**Figure 14.** The continuous authentication phase with identity anonymity.

In the static authentication phase and the continuous authentication phase of the modified protocol, the sensor node uses the random numbers  $r_1$  and  $r_2$  to compute an anonymous identity  $AID_{SN} = H(H(ID_{SN} || r_1) \oplus H(ID_{SN}))$  and  $AID_{SN} = H(H(ID_{SN} || r_2) \oplus H(ID_{SN}))$ , respectively. In the modified protocol, the anonymous identity  $AID_{SN}$  is used during data transmission instead of the original identity  $ID_{SN}$ . After the gateway receives  $AID_{SN}$ , it derives the identity of a sensor node  $ID_{SN}$  by generating a tentative  $AID'_{SN} = H(H(ID_{SN} || r_1) \oplus H(ID_{SN}))$  for each sensor identity  $ID_{SN}$  in the database and evaluating the equivalence between  $AID'_{SN}$  and the received  $AID_{SN}$ .

## 6. Conclusions

In order to improve the social welfare of future human life through data analytics and the interaction interface between humans and their environment, new information systems with real-time sensed data from individuals and environments need to be developed and deployed in modern cities. To facilitate implementation of these systems, IoT-based devices such as sensing devices and intermediary gateways have to be deployed in different environments to form the IoT infrastructure. One of the fundamental security issues for IoT infrastructure is how to mutually authenticate both communicating peers before a sensing device transmits sensed data to an intermediary gateway. As sensed data will be transmitted to gateways periodically, a lightweight authentication protocol for device-to-device communication is indeed required to preserve device energy and extend the battery lifecycle correspondingly.

To accomplish this goal, a lightweight continuous authentication protocol for IoT infrastructure is proposed. The proposed protocol has several characteristics. First, the concept of valid authentication time period and continuous authentication are introduced and adopted. Second, the token technique and dynamic features of IoT devices, i.e., the remaining battery capacity, are adopted to quickly authenticate communicating parties in each session. Third, the computation process during an authentication session does not use encryption/decryption operations in order to reduce time consumption of these computation operations. In addition, a security analysis and a performance analysis are conducted for the proposed protocol to evaluate its security strength and its competitiveness in terms of time consumption for mutual authentication in a session. The proposed protocol can also be extended in two aspects: adopting the protocol implementation option of gateway-initialized request and adding the feature of identity anonymity onto sensing devices. For the future work, inventing a more accurate model of battery energy consumption and discovering more dynamic device features are two practical and interesting challenges for researchers.

**Acknowledgments:** The authors acknowledge the support from Taiwan Information Security Center (TWISC) and Ministry of Science and Technology, Taiwan, under the Grant Numbers MOST 106-3114-E-001-003, MOST 105-2221-E-011-070-MY3, MOST 105-2221-E-011-080-MY3, and MOST 106-2218-E-011-003.

**Author Contributions:** Yo-Hsuan Chuang, Cheng-Ying Yang and Nai-Wei Lo conceived and designed the protocol and applicable scenarios; Yo-Hsuan Chuang and Nai-Wei Lo analyzed protocol security; Cheng-Ying Yang and Ssu-Wei Tang analyzed protocol performance; Yo-Hsuan Chuang, Nai-Wei Lo, Cheng-Ying Yang and Ssu-Wei Tang wrote and revised the article.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Perera, C.; Liu, C.H.; Jayawardena, S. The Emerging Internet of Things Marketplace from an Industrial Perspective: A Survey. *IEEE Trans. Emerg. Top. Comput.* **2015**, *3*, 585–598. [[CrossRef](#)]
2. Coetzee, L.; Eksteen, J. The Internet of Things—Promise for the Future? An Introduction. In Proceedings of the IST-Africa Conference, Gaborone, Botswana, 11–13 May 2011; pp. 1–9.
3. Internet of Things (IoT) Cybersecurity Colloquium, National Institute of Standards and Technology (NIST), NISTIR 8201. Available online: <https://nvlpubs.nist.gov/nistpubs/ir/2017/NIST.IR.8201.pdf> (accessed on 21 May 2017).
4. Al-Fuqaha, A.; Guizani, M.; Mohammadi, M.; Aledhari, M.; Ayyash, M. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 2347–2376. [[CrossRef](#)]
5. Mahmoud, R.; Yousuf, T.; Zualkeman, I. Internet of things (IoT) Security: Current Status, Challenges and Prospective Measures. In Proceedings of the 2015 10th International Conference for Internet Technology and Secured Transactions (ICITST), London, UK, 14–16 December 2015; pp. 336–341.
6. Shivraj, V.L.; Rajan, M.A.; Singh, M.; Balamuralidhar, P. One Time Password Authentication Scheme Based on Elliptic Curves for Internet of Things (IoT). In Proceedings of the 2015 5th National Symposium on Information Technology: Towards New Smart World (NSITNSW), Riyadh, Saudi Arabia, 17–19 February 2015; pp. 1–6.



7. Abomhara, M.; Kjøien, G.M. Security and Privacy in the Internet of Things: Current Status and Open Issues. In Proceedings of the 2014 International Conference on Privacy and Security in Mobile Systems (PRISMS), Aalborg, Denmark, 11–14 December 2014; pp. 1–8.
8. Alqassem, I.; Svetinovic, D. A Taxonomy of Security and Privacy Requirements for the Internet of Things (IoT). In Proceedings of the 2014 IEEE International Conference on Industrial Engineering and Engineering Management, Bandar Sunway, Malaysia, 9–12 December 2014; pp. 1244–1248.
9. Traore, I.; Woungang, I.; Nakkabi, Y.; Obaidat, M.S.; Ahmed, A.A.E.; Khalilian, B. Dynamic Sample Size Detection in Learning Command Line Sequence for Continuous Authentication. *IEEE Trans. Syst. Man Cybern.* **2012**, *42*, 1343–1356. [[CrossRef](#)] [[PubMed](#)]
10. Mondal, S.; Bours, P. Continuous Authentication in a Real World Settings. In Proceedings of the 2015 Eighth International Conference on Advances in Pattern Recognition (ICAPR), Kolkata, India, 4–7 January 2015; pp. 1–6.
11. Buduru, A.B.; Yau, S.S. An Effective Approach to Continuous User Authentication for Touch Screen Smart Devices. In Proceedings of the 2015 IEEE International Conference on Software Quality, Reliability and Security (QRS), Vancouver, BC, Canada, 3–5 August 2015; pp. 219–226.
12. Mondal, S.; Bours, P. Continuous Authentication and Identification for Mobile Devices: Combining Security and Forensics. In Proceedings of the 2015 IEEE International Workshop on Information Forensics and Security (WIFS), Rome, Italy, 16–19 November 2015; pp. 1–6.
13. Brocardo, M.L.; Traore, I.; Woungang, I. Toward a Framework for Continuous Authentication Using Stylometry. In Proceedings of the 2014 IEEE 28th International Conference on Advanced Information Networking and Applications, Victoria, BC, Canada, 13–16 May 2014; pp. 106–115.
14. Bamasag, O.O.; Youcef-Toumi, K. Towards Continuous Authentication in Internet of Things Based on Secret Sharing Scheme. In Proceedings of the WESS'15: Workshop on Embedded Systems Security, Amsterdam, The Netherlands, 4–9 October 2015; pp. 1–8.
15. Bormann, C.; Ersue, M.; Keranen, A. Terminology for Constrained-Node Networks. RFC 7228, Internet Engineering Task Force (IETF). 2014. Available online: <https://tools.ietf.org/html/rfc7228> (accessed on 20 May 2017).
16. Sethi, M.; Arkko, J.; Keranen, A.; Back, H. Practical Considerations and Implementation Experiences in Securing Smart Object Networks. Draft-Ietf-Lwig-Crypto-Sensors-06. 2018. Available online: <https://tools.ietf.org/pdf/draft-ietf-lwig-crypto-sensors-06.pdf> (accessed on 20 May 2017).
17. Atzori, L.; Iera, A.; Morabito, G.; Giacomo, M. The Internet of Things: A Survey. *Comput. Netw.* **2010**, *54*, 2787–2805. [[CrossRef](#)]
18. Khemissa, H.; Tandjaoui, D. A Lightweight Authentication Scheme for E-Health Applications in the Context of Internet of Things. In Proceedings of the 2015 9th International Conference on Next Generation Mobile Applications, Services and Technologies, Cambridge, UK, 9–11 September 2015; pp. 90–95.
19. Khemissa, H.; Tandjaoui, D. A Novel Lightweight Authentication Scheme for Heterogeneous Wireless Sensor Networks in the Context of Internet of Things. In Proceedings of the 2016 Wireless Telecommunications Symposium (WTS), London, UK, 18–20 April 2016; pp. 1–6.
20. Mahalle, P.N.; Prasad, N.R.; Prasad, R. Threshold Cryptography-based Group Authentication (TCGA) Scheme for the Internet of Things (IoT). In Proceedings of the 2014 4th International Conference on Wireless Communications, Vehicular Technology, Information Theory and Aerospace & Electronic Systems (VITAE), Aalborg, Denmark, 11–14 May 2014; pp. 1–5.
21. Porambage, P.; Schmitt, C.; Kumar, P.; Gurtov, A.; Ylianttila, M. Two-phase Authentication Protocol for Wireless Sensor Networks in Distributed IoT Applications. In Proceedings of the 2014 IEEE Wireless Communications and Networking Conference (WCNC), Istanbul, Turkey, 6–9 April 2014; pp. 2728–2733.
22. Krawczyk, H.; Bellare, M.; Canetti, R. HMAC: Keyed-Hashing for Message Authentication. RFC 2104, Internet Engineering Task Force (IETF). 1997. Available online: <https://www.rfc-editor.org/rfc/rfc2104.txt> (accessed on 26 July 2017).
23. Rescorla, E.; Modadugu, N. Datagram Transport Layer Security Version 1.2. RFC 6347, Internet Engineering Task Force (IETF). 2012. Available online: <https://www.rfc-editor.org/rfc/rfc6347.txt> (accessed on 26 July 2017).
24. Kothmayr, T.; Schmitt, C.; Hu, W.; Brüning, M.; Carle, G. DTLS Based Security and Two-way Authentication for the Internet of Things. *Ad Hoc Netw.* **2013**, *11*, 2710–2723. [[CrossRef](#)]

25. Goh, E.J. Encryption Schemes from Bilinear Maps. Ph.D. Thesis, Stanford University, Stanford, CA, USA, 2007.
26. Paillier, P. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Advances in Cryptology—EUROCRYPT '99*; Springer: Berlin/Heidelberg, Germany, 1999; pp. 223–238.
27. Advanced Encryption Standard (AES), Federal Information Processing Standards Publication 197, National Institute of Standards and Technology (NIST). Available online: <http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf> (accessed on 26 July 2017).
28. Kumar, P.; Gurtov, A.; Iinatti, J.; Ylianttila, M.; Sain, M. Lightweight and Secure Session-Key Establishment Scheme in Smart Home Environments. *IEEE Sens. J.* **2016**, *16*, 254–264. [[CrossRef](#)]
29. Gope, P.; Hwang, T. Untraceable Sensor Movement in Distributed IoT Infrastructure. *IEEE Sens. J.* **2015**, *15*, 5340–5348. [[CrossRef](#)]
30. Kawamoto, Y.; Nishiyama, H.; Kato, N.; Shimizu, Y.; Takahara, A.; Jiang, T. Effectively Collecting Data for the Location-Based Authentication in Internet of Things. *IEEE Syst. J.* **2015**, *11*, 1403–1411. [[CrossRef](#)]
31. Shimshon, T.; Moskovitch, R.; Rokach, L.; Elovici, Y. Continuous Verification Using Keystroke Dynamics. In Proceedings of the 2010 International Conference on Computational Intelligence and Security (CIS), Nanning, China, 11–14 December 2010; pp. 411–415.
32. Shen, C.; Cai, Z.; Guan, X. Continuous Authentication for Mouse Dynamics: A Pattern-growth Approach. In Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012), Boston, MA, USA, 25–28 June 2012; pp. 1–12.
33. Bailey, K.O.; Okolica, J.S.; Peterson, G.L. User Identification and Authentication Using Multi-modal Behavioral Biometrics. *Comput. Secur.* **2014**, *43*, 77–89. [[CrossRef](#)]
34. Niinuma, K.; Park, U.; Jain, A.K. Soft Biometric Traits for Continuous User Authentication. *IEEE Trans. Inf. Forensics Secur.* **2010**, *5*, 771–780. [[CrossRef](#)]
35. Mock, K.; Hoanca, B.; Weaver, J.; Milton, M. Real-time Continuous Iris Recognition for Authentication Using an Eye Tracker. In Proceedings of the 2012 ACM Conference on Computer and Communications Security, Raleigh, NC, USA, 16–18 October 2012; pp. 1007–1009.
36. Peng, G.; Zhou, G.; Nguyen, D.T.; Qi, X.; Yang, Q.; Wang, S. Continuous Authentication with Touch Behavioral Biometrics and Voice on Wearable Glasses. *IEEE Trans. Hum. Mach. Syst.* **2017**, *47*, 404–416. [[CrossRef](#)]
37. Zhou, L.; Su, C.; Chiu, W.; Yeh, K.H. You Think, Therefore You Are: Transparent Authentication System with Brainwave-oriented Bio-features for IoT Networks. *IEEE Trans. Emerg. Top. Comput.* **2017**. [[CrossRef](#)]
38. Seitz, L.; Gerdes, S.; Selander, G.; Mani, M.; Kumar, S. Use Cases for Authentication and Authorization in Constrained Environments. RFC 7744, Internet Engineering Task Force (IETF). 2016. Available online: <https://tools.ietf.org/html/rfc7744> (accessed on 20 May 2017).
39. Scyther. Available online: <https://www.cs.ox.ac.uk/people/cas.cremers/scyther/> (accessed on 20 May 2017).
40. Gavin, L. A Hierarchy of Authentication Specifications. In Proceedings of the 10th IEEE Workshop on Computer Security Foundations, Rockport, MA, USA, 10–12 June 1997.
41. Cremers, C.J.F.; Mauw, S.; Vink, E.P. Injective Synchronisation: An Extension of the Authentication Hierarchy. *Theor. Comput. Sci.* **2006**, *367*, 139–161. [[CrossRef](#)]
42. Pereira, G.C.C.F.; Alves, R.C.A.; da Silva, F.L.; Azevedo, R.M.; Albertini, B.C.; Margi, C.B. Performance Evaluation of Cryptographic Algorithms over IoT Platforms and Operating Systems. *Secur. Commun. Netw.* **2017**, *2017*, 2046735. [[CrossRef](#)]
43. Yeh, K.H.; Su, C.; Choo, K.R.; Chiu, W. A Novel Certificateless Signature Scheme for Smart Objects in the Internet-of-Things. *Sensors* **2017**, *17*, 1001. [[CrossRef](#)] [[PubMed](#)]

