Abstract: EnOcean, a commonly used control protocol in smart lighting systems, provides authentication, as well as message integrity verification services, and can resist replay attack and tamper attack. However, since the device identity information transmitted between sensors in smart lighting control systems is easily accessible by malicious attackers, attackers can analyze users’ habits based on the intercepted information. This paper analyzed the security of the EnOcean protocol using a formal analysis method based on the colored Petri net (CPN) theory and the Dolev–Yao attacker model and found that the protocol did not anonymize the device identity information and did not have a communication key update mechanism, so an attacker could easily initiate a key compromise impersonation attack (KCIA) after breaking the pre-shared communication key. To address the above security issues, this paper proposed an EnOcean-A protocol with higher security based on the EnOcean protocol. The EnOcean-A protocol introduced a trusted third-party server to send communication keys to communication devices because devices must obtain different communication keys from the trusted third-party server each time they communicated. Thus, this protocol could resist a KCIA and achieve forward security. Meanwhile, the device identity information was anonymized using a homomorphic hash function in the EnOcean-A protocol, and the dynamic update mechanism of the device identity information was added so that an attacker could not obtain the real identity information of the device. Finally, the formal analysis of the EnOcean-A protocol showed that the new protocol could resist a KCIA and ensure the anonymity and untraceability of the communication device, which had higher security compared with the EnOcean protocol.

Keywords: smart lighting control; EnOcean protocol; colored Petri net; Dolev–Yao attacker model

1. Introduction

The smart lighting control system, a novel internet of things (IoT) product, has emerged quickly in the industrial and commercial markets due to the rapid development of wireless communication technology and sensor technology [1]. Not only can smart lighting control systems offer lighting for industrial environments, but they can also give homeowners more intelligent living spaces. For instance, a lighting control system can activate a fixed-position lighting device in accordance with a user’s location when the user opens a door and enters a room [2].

The application process for the smart lighting control system must gather data about users’ everyday lives, including their location and entry and leave times, in order to properly perform. Private information about users’ movements and positions is gathered by sensing devices to offer more intelligent service experiences [3,4]. However, the majority of these sensor devices communicate data over wireless networks. Wireless transmission channels’ openness makes it simple for malicious attackers to intercept and tamper with the sent data. In order to determine if a smart lighting control system can guarantee security, it is crucial to ensure that personal data is not compromised throughout the IoT application process. Neither the current nor the updated lighting control protocol can, however, provide a greater level of data privacy protection.
One of the protocols used to ensure secure communication for IoT devices is Zigbee [5]. Farha et al. [6] proposed an enhanced ZigBee protocol scheme employing an enhanced timestamp mechanism that is provided because the security issue that the ZigBee protocol is vulnerable to is replay attack. Khanji et al. [7] suggested updating the key management and key update mechanisms to address the security issues with the ZigBee protocol, including denial of service (DOS), flooding, desynchronization, and wormhole attacks. In smart lighting control systems, the Bluetooth protocol is frequently employed [8,9]. Lonzetta et al. [10] and Zhang et al. [11] analyzed smart home devices employing the Bluetooth protocol that were vulnerable to man-in-the-middle (MITM) attacks, wormhole attacks, and sniffer attacks. Long-range wide-area network (LoRaWANs) are often used in lighting control systems due to their low power consumption and capacity for long-distance communication [12–14]. A new key management protocol has been proposed in [15–19] to address the issues of key management and key update in LoRaWAN protocol and to fight against key compromise and replay attack. Nutun et al. [20] discovered security issues in the LoRaWAN protocol, such as synchronous attack and MITM attack, and proposed security approaches to improve the protocol’s security. Tsai et al. [21] suggested a novel session key generation approach by integrating elliptic curve cryptography (ECC) with an advanced encryption standard (AES) encryption method to generate a secure session key between devices in order to address the security flaws between application layers in the LoRaWAN protocol. According to the aforementioned paper, existing and enhanced smart lighting control protocols primarily offer security protection against MITM attacks, DOS attacks, and replay attacks but do not offer measures to anonymize sensor device information.

EnOcean Alliance proposed the EnOcean wireless transmission system [22]. In March 2012, the International Electrotechnical Commission (IEC) approved the EnOcean wireless communication standard as an international standard (ISO/IEC 14543-3-10). EnOcean is a wireless communication system with low power requirements, ad hoc networking, strong security, and reliability. All sensors, wireless switches, and other equipment that support the EnOcean protocol do not require batteries or other power sources. This reduces the need for complex equipment maintenance in the future, especially for smart lighting control systems [23].

Since smart lighting control systems are widely used in life, providing people with a more convenient and smarter life experience, most of the existing intelligent lighting control systems are able to provide message encryption, identity authentication, and message integrity verification services. However, because the device identity information transmitted between sensors in an intelligent lighting control system is easily accessible by malicious attackers, attackers can analyze a user’s living habits based on the intercepted information. Through the related literature, it can be found that most of the existing smart lighting control protocols cannot anonymize the device identity, so this paper proposes a protocol with higher security based on the EnOcean protocol. The main contributions of this paper are as follows:

- In this paper, we use CPN theory and the Dolev–Yao attacker model to formally analyze the EnOcean protocol. Through a security analysis, we find that the EnOcean protocol is able to provide authentication and message integrity verification services, but the EnOcean protocol does not anonymize the identity information of the communication devices, so attackers can analyze a user’s living habits based on the eavesdropped device identity information. The formal analysis reveals that the EnOcean protocol cannot resist a KCIA.
- To address the above security issues, this paper proposes the EnOcean-A protocol with higher security. The EnOcean-A protocol introduces a trusted third-party server to distribute communication keys to communication devices, and the devices can obtain new communication keys from the trusted third-party server before communication, which can resist KCIA and provide forward security. The EnOcean-A protocol uses a homomorphic hash function to anonymize the device identity information and adds a dynamic update mechanism for device identity information so that an attacker cannot
obtain the real identity information of the device, thus providing unlinkability of the device identity information.

- Finally, the security of EnOcean-A protocol is analyzed by a formal analysis method, and it is found that the EnOcean-A protocol has higher security compared with the EnOcean protocol.

2. Related Works

Several distinct anonymous approaches have been proposed for IoT devices. The traceability of device identities in smart home systems enables attackers to track device activity and infer users’ daily routines. Sadri et al. [24] proposed a solution that uses two-factor authentication to authenticate user identification and symmetric key encryption to transmit data. Banerjee et al. [25] presented a simple, anonymous authentication mechanism using the user’s smart card, password, and unique biometric data. Fakroon et al. [26] proposed a unique user authentication mechanism that utilized physical information. User information must be saved on the device for the three physical-factor-based authentication mechanisms described above. They are not suitable for protocols used only for authentication between devices. Without a trusted third-party server involved, Hajian et al. [27] proposed a key agreement technique for mutual authentication between two devices. In this protocol, the registration and key update phases were conducted in a common channel, in addition to the identity authentication phase. Li et al. [28] suggested an anonymous authentication and key agreement mechanism (AAKA). To accomplish authentication and privacy protection, this protocol made use of dynamic sequence numbers and dynamic random numbers. The proposed protocol worked well for identity authentication between a user, sensor node, and gateway and did not rely on a trusted third-party server. Lightweight, anonymous protocols cannot be developed without a third-party server since the computing capabilities of lighting sensor devices do not allow them to calculate keys and complicated encryption algorithms. A lightweight, secure smart home protocol based on mutual anonymous authentication and device key agreement was proposed in Hasan et al. [29] and Banerjee et al. [30]. However, only the user, sensor, and gateway nodes’ identities could be authenticated using this type of approach. Rasheed et al. [31] presented a zero-knowledge proof-based authentication mechanism. However, this approach could be used with a multicast environment’s identity authentication protocol. It was not necessary to authenticate between two nodes. Shuai et al. [32] proposed an effective anonymous authentication method based on ECC suggested to be employed in smart home environments. To resist against replay attack and prevent clock synchronization, the scheme employed random numbers. This method utilized the ECC public key encryption standard, which has certain performance requirements for devices. An anonymous security framework (ASF) appropriate for a smart home environment was suggested in Kumar et al. [33]. This framework provided device anonymity and effective authentication and key agreement scheme. The device identity in this method, however, was static and could not be utilized to create untraceable communication devices. Table 1 provides a comparative analysis of the above articles.
Table 1. Comparison of related works.

| Author(s) | Main Contribution | Drawbacks |
|-----------|-------------------|-----------|
| Sadri et al. [24] | Proposed a two-factor authentication protocol | These three authentication schemes required user’s physical information and were not applicable to device-to-device authentication protocol |
| Banerjee et al. [25] | Presented a three-factor authentication scheme | |
| Fakroon et al. [26] | Proposed a new authentication scheme that combined physical context | Neither of these protocols used a third-party server. The computational performance of lightweight sensor devices could not perform the key computation and complex encryption algorithms, so such designs were not suitable for communication between lightweight sensor devices with limited computational power and storage space |
| Hajian et al. [27] | Suggested an authentication protocol between two devices | |
| Li et al. [28] | Proposed an anonymous authentication and key negotiation protocol | These two schemes were only applicable for authentication between three points: the user, sensor node, and gateway node. They were not applicable to device-to-device authentication services |
| Hasan et al. [29] | Proposed a lightweight, secure smart home protocol based on mutual anonymous authentication and key negotiation of devices | |
| Banerjee et al. [30] | Suggested a more secure and robust authentication scheme | |
| Rasheed et al. [31] | Proposed a zero-knowledge proof-based authentication mechanism | This approach could be used with a multicast environment’s identity authentication protocol. It was not necessary to authenticate between two nodes |
| Shuai et al. [32] | Proposed an efficient, anonymous authentication scheme based on ECC for smart home environments | The ECC public key encryption scheme was used in this scheme, which has a certain demand on the performance of the device. However, the computational performance of lightweight sensor devices could not accomplish the key computation and the complex encryption algorithm |
| Kumar et al. [33] | Suggested an anonymous security framework for smart home environments | The device information in this scheme did not change dynamically and did not provide untraceability of communication devices |

Hofer-Schmitz [34] presented the first formal analysis method for the EnOcean protocol and discovered design flaws in the EnOcean protocol through the formal analysis method. Based on EnOcean’s security documentation, the authors used the pi calculus formal analysis method and the ProVerif tool to test the security of the EnOcean protocol. The results showed that, although the EnOcean protocol could guarantee its stated security, it did not encrypt the identity information of the sending device during the teach-in phase, so an attacker could send an intercepted message to another receiving device. The author proposed a solution to the above problem. However, this formal analysis did not reveal that the EnOcean protocol was insecure against KCIA and that the protocol did not anonymize the identity information of the device during operation.

The literature mentioned above revealed that the majority of the current anonymous authentication mechanisms and schemes have the following issues: (a) The IoT terminal devices may not have enough computing power to support necessary activities if they rely on ECC public key encryption. (b) While ignoring the anonymous authentication between sensors, they focus primarily on the three-party authentication between the user, the sensor, and the gateway. (c) Since there is no method for dynamic identification information modification and only device information may be anonymized, sensor device untraceability is not achievable. Moreover, the existing security analysis of the EnOcean protocol did not suggest that there was no device anonymization operation in the EnOcean protocol. Under the condition that the sensor nodes are limited in power energy, communication capability, and computational and storage capacity of the chip, this paper introduces a
trusted third-party server to send communication keys to the communication nodes and provides anonymization for devices using only homomorphic hash functions.

3. Preliminary Knowledge

3.1. EnOcean Protocol

The EnOcean protocol includes two phases: teach-in and authentication [34]. For wireless network authentication, a teach-in phase must be performed so the receiver can determine whose device sent the message. The EnOcean protocol employs a 32-bit chip ID as the device identification [35]. Devices are able to determine if a message needs to be processed based on the sender ID learned during the teach-in phase. The pre-shared key is typically written on a sticker on the sender module. EnOcean uses variable AES encryption and cipher-based message authentication codes to verify message integrity.

The EnOcean protocol is shown below, and Table 2 shows specific symbols.

| Symbol   | Definition                                      |
|----------|-------------------------------------------------|
| ID_A     | Chip ID of sending device                       |
| ID_B     | Chip ID of receiving device                     |
| A, B     | Sender A, Receiver B                            |
| RLC      | Rolling code                                    |
| PSK      | Pre-shared key                                  |
| Key      | Session key                                     |
| RND      | Random number                                   |
| Payload  | Payload of telegram                             |
| CMAC, CMAC' | Cipher-based message authentication code     |
| Enc(·)   | Encryption function                             |

Step 1: A and B switch on the learn mode. A generates the RLC and Key and encrypts them using the PSK. A transmits ID_A, ID_B, and the encrypted data to B. B confirms whether the message is received after receiving it. B discards the message if it has already been received. Otherwise, the teach-in phase is over, and the Key and RLC are saved.

Step 2: For the authentication phase, A sends a challenge message (ID_A, ID_B) to B.

Step 3: After receiving the message, B generates an RND and sends (ID_A, ID_B, RND) to A.

Step 4: A calculates the CMAC value of the Payload and RND using the Key after receiving the message. The Payload and CMAC are encrypted with the Key. ID_A, ID_B, and the encrypted data are sent from A to B.

Step 5: B receives the message and uses the Key to decrypt it in order to acquire the Payload and CMAC. B also calculates the CMAC' using the Payload and RND. A is successfully authenticated, and B saves the Payload if the CMAC = CMAC'. In Figure 1, the specific procedure is displayed.
3.2. Homomorphic Hash Function

Homomorphic hash function refers to a hash algorithm with homomorphism [36,37]. If $Z_p$ is a finite field of positive integers $p$ and the vector $g$ is a $1 \times 1$ row vector, then each entry is a member of $Z_p$ of order $q$, and $l$ is a positive integer. The elements $m_i, m_j, i, j \in \{1, 2, \ldots, n\}$ in the data block $F = \{m_1, m_2, \ldots, m_n\}$ are all elements of the vector $Z_p$, and $H(\cdot)$ is homomorphic hash function, while $t$ is a positive integer:

$$H(m_i) = \prod_{t=1}^{l} g_{i}^{m_{t,i}} \mod p$$

Then, the hash of the addition of two vectors $m_i$ and $m_j$ is as follows:

$$H(m_i + m_j) = \prod_{t=1}^{l} g_{i}^{m_{t,i} + m_{t,j}} \mod p = \prod_{t=1}^{l} g_{i}^{m_{t,i}} \cdot g_{t}^{m_{t,j}} \mod p = \prod_{t=1}^{l} g_{i}^{m_{t,j}} \mod p \times \prod_{t=1}^{l} g_{i}^{m_{t,i}} \mod p = H(m_i) \times H(m_j)$$

It can be seen that the hash algorithm is homomorphic. Homomorphic hash functions also have a collision avoidance property. For a homomorphic hash function, for any two pieces of information $m_1, m_2$, and for real numbers $w_1, w_2$, the homomorphism has the following:

$$H(w_1 m_1 + w_2 m_2) = H(m_1)^{w_1} H(m_2)^{w_2}$$
The attacker does not exist in the probability polynomial algorithm, which can obtain $(m_1, m_2, m_3, w_1, w_2)$, where $m_3 \neq w_1m_1 + w_1m_2$, and $m_3$ is a piece of information:

$$H(m_3) = H(m_1)^{w_1} + H(m_2)^{w_2}$$

3.3. CPN Modeling Tool

CPN is a modeling language [38,39] that successfully combines the standard meta language (ML) for net systems with the Petri net theory. Common systems can be handled by CPN, and it can simulate and analyze concurrent systems. A CPN model of a system is typically executable and can describe both the current state of the system and efforts to change it. This event is a transition in the CPN model. CPN Tools is essential in dealing with systems such as concurrency, interactivity, and synchrony due to its advantages in modeling and verifying model systems.

3.4. Dolev–Yao Attacker Model

The Dolev–Yao attacker model proposed by D. Dolev and A. Yao [40] formally characterized the behavior of an attacker and offered a mathematical model for testing public key cryptography protocol. Researchers can benefit by focusing on the inherent security qualities of protocols rather than the security of cryptographic algorithms by discussing the security properties of protocols based on the assumption that the cryptosystem is “perfect”.

The attacker has the following functions:

- The attacker has access to the open channel. The data transmitted between entities while the protocol is executing can be intercepted, altered, and replayed by the attacker;
- The attacker can encrypt, decrypt, split, and combine the original message through the known content to forge the message content;
- Using the identity of the sensor that was eavesdropped, the attacker can determine whether it was delivered by the same device;
- The attacker can break the long-used key and simulate the device to launch an attack using this key.

4. EnOcean Protocol HCPN Modeling

4.1. EnOcean Protocol Color Set Definition

The color set is established for the four messages exchanged between the receiver and the sender. First of all, communication information is composed of four pieces of meta-information: ID, RANDOM, KEY, and PAYLOAD. ID indicates the identity of the device chip, RANDOM indicates a random number, PAYLOAD indicates the payload sent by the sender, and KEY indicates the encryption key. The MSG1 type indicates that teach-in messages are sent from the sender to the receiver. The MSG2 type indicates that the sender sends an identity authentication request to the receiver. The MSG3 type indicates that the receiver sends a random number for authentication to the sender. The MSG4 type indicates that the sender sends the authentication calculation results to the receiver. The specific color set definitions are shown in Table 3.

| Key Elements | Color Set Definition |
|--------------|----------------------|
| ID           | colset ID = with ida \| idb; |
| RANDOM       | colset RANDOM = with rlc \| rndb \| rndb’ \| none; |
| KEY          | colset KEY = with S \| SA; |
| PAYLOAD      | colset PAYLOAD = STRING; |
| MSG1         | colset MSG1 = record ida:ID \times idb:ID \times enc:ENC_IN1; |
| MSG2         | colset MSG2 = record ida:ID \times idb:ID; |
| MSG3         | colset MSG3 = record ida:ID \times idb:ID \times rnd:Random; |
| MSG4         | colset MSG4 = record ida:ID \times idb:ID \times enc2:ENC2; |
4.2. Formal Analysis of EnOcean Protocol

In this study, a top-down formal analysis of the EnOcean protocol was modeled using CPN Tools. The protocol submodule was established after the protocol top-level model was defined. The transition was represented by a rectangle, the place was represented by an ellipse, and the substitution transition was represented by a double-line rectangular transition, indicating that the transition had more detailed submodels.

The EnOcean protocol’s top model had 12 places and 7 substitution transitions. The substitution transition of Teach-in represented the unique procedure of the sender providing teach-in information to the receiver. The substitution transition of Learn represented how teach-in signals were received and analyzed by the receiver. The substitution transition of Connection represented the procedure through which the sender sent an authentication request and received a random number. The substitution transition of Authentication represented the procedure where the receiver compared the CMAC values supplied by the sender. Figure 2 illustrates the specifics.

A detailed explanation of the partial substitution transition follows. The sub-model that replaced the Teach-in transition is shown in Figure 3. The rlc and session key were first sent by the transition of Combination to the transition of Encry1, which then encrypted the message using the pre-shared key and transmitted it to the transition of Teach-in. Together with the sender and receiver’s identities, the transition of Teach-in delivered the encrypted message to the place Send_MSG1.
Figure 4 is the internal model of the substitution transition Learn. First, the transition of Split split the received MSG1 information, stored the identity information of the sender and receiver in places ID_A and ID_B, and sent the remaining information to the transition of Decryption. The transition of Decryption used the pre-shared key to decrypt the received message. If the key was different, the decryption failed. Otherwise, the decryption message was sent to the transition of Collection. The transition of Collection sent the RLC in the decryption message to the place Store_RLC for saving and the session key to the place Session_Key'.

Figure 5 shows the internal model of the substitution transition of Authentication. The place Related_Message sent a message to the transition of Split_Related_Message. The transition of Split_Related_Message split the received message and sent the random number to the transition of Connection_CMAC_IN and the session key to the transition of Compute_CMAC. The transition of Connection_CMAC_IN combined random numbers and authentication messages to the transition of Compute_CMAC. The transition of Compute_CMAC computed the CMAC value of the received message using the session key and sent the CMAC to the transition of Compute_ENC_IN. The transition of Encry_ENC_IN2 encrypted the received message using the session key and sent the encrypted message to the transition of Combination_MSG4. The transition of Combination_MSG4 sent the received encrypted message and sender and receiver identities to the place Send_MSG4.
Figure 5. Substitution transition authentication internal model.

Figure 6 shows the internal model of the substitution transition of Authentication. The transition of Split_MSG4 split the received message, saved the identity of the sender and receiver to places ID_A and ID_B, and sent the encrypted message to the transition of Decry_ENC2. The transition of Decry_ENC2 used the session key to decrypt the encrypted message. If the encrypted message was not encrypted with the session key, the decryption failed. Otherwise, the decryption message was sent to the transition of Split_ENC_IN2. The transition of Split_ENC_IN2 split and decrypted the message, saved the CMAC value in place Store_CMAC, and saved the authentication message in place Store_Payload. Place Related_Message' sent a message to place Split_R_MSG. Place Split_R_MSG split the received message and sent the random number to the transition of Combination_CMAC_IN. The transition of Combination_CMAC_IN sent random numbers and authentication messages to the transition of Compute_CMAC, which computed the CMAC value using the session key and sent the result to the transition of Compare_CMAC. The transition of Compare_CMAC compared the received CMAC value with the calculated CMAC value. If the values were equal, it indicated that the receiver authenticated the sender successfully and saved the authentication success message to the place Auth_Success. Otherwise, the authentication failure message was saved to the place Auth_Fail.
4.3. EnOcean Protocol Consistency Analysis

State-space analysis methods were used to analyze the EnOcean protocol's conformance. The numbers of nodes and arcs in a state-space were the same as the numbers of strongly connected nodes and strongly connected arcs, according to an analysis of the state-space results in Table 4. This finding shows that the established EnOcean protocol model did not result in state cycles and that all the state nodes were reachable. One dead node indicated that the receiver handled all the requests, allowing the model to function as intended.

Table 4. State-space analysis of EnOcean protocol model.

| Type                  | Number |
|-----------------------|--------|
| State-Space Nodes     | 1380   |
| State-Space Arcs      | 3582   |
| Scc Graph Nodes       | 1380   |
| Scc Graph Arcs        | 3582   |
| Dead Marking          | 1      |
| Dead Transition       | 0      |

4.4. EnOcean Protocol Security Evaluation Based on Dolev–Yao Attacker Model

The replay attack, tamper attack, and KCIA were introduced in the network transport layer of the EnOcean protocol model. The attacker could analyze whether the information was sent by the same device based on the intercepted device identity information, and when the attacker obtained the pre-shared key, he could decrypt the information transmitted between devices for a second time and tamper with the transmitted information. The places and transitions marked in blue in Figure 7 simulate a KCIA; the transitions T0 and t0 intercept the information during the protocol transmission; the places P1 and P3 are able to store the decomposed and to-be-decomposed information; and the places P1, P2, Atom, P6, P5, IDA, IDB, RND, p3, p8, p9, and Atom store the atomic information. Transitions T2, T2', t1, and t1' indicate the decryption of intercepted messages with a known key. Transitions T5 and t11 indicate the reassembly of the decomposed messages. The red part in Figure 7 indicates that a tamper attack was simulated, and the tamper attack is launched by transition T3'. The purple part in Figure 7 indicates that a replay attack is simulated, and the replay attack is launched through the transition of Reply_MSG1'.

Figure 6. Substitution transition authentication internal model.
Figure 7. EnOcean protocol attacker model.
4.5. **EnOcean Protocol Security Attribute Verification Analysis**

All state nodes in the attacker model of this protocol were reachable, as shown by the state space report on the attacker model in Table 5. The numbers of state space nodes and directed arcs were equal to the numbers of strongly connected nodes and strongly connected arcs.

| Type                  | Tamper Attack | Replay Attack | KCIA  |
|-----------------------|---------------|---------------|-------|
| State Space Nodes     | 7380          | 131           | 4537  |
| State Space Arcs      | 24,672        | 280           | 12,318|
| Scc Graph Nodes       | 7380          | 131           | 4537  |
| Scc Graph Arcs        | 24,672        | 280           | 12,318|
| Dead Markings         | 1             | 1             | 1     |
| Dead Transitions      | 0             | 23            | 0     |

As indicated in Table 5, an analysis of the state-space data from the EnOcean protocol added to the attacks. A tamper attack was implemented in the protocol’s transport layer of MSG3. Given that MSG3 messages were sent in plaintext, an attacker could directly alter random integers in the message. However, the CMAC algorithm was employed by this protocol. The CMAC value was determined by the sender using random numbers and authentication messages, and the matching CMAC value was determined by the receiver using the same CMAC algorithm. The receiver concluded that the identity authentication of the sender failed when it compared the CMAC values and discovered that the two CMAC values were different. The query statement found the failed value in the place Auth_Fail on the receiver, indicating that identity authentication failed. Figure 8 displays the details.

In MSG1, a replay attack was added at the transport layer. When an attacker launched a replay attack, the receiver checked to see whether the RLC in the message was stored in the MSG1 messages. The receiver recognized that a replay attack had taken place if the RLC was already present and stopped the further actions. Replay attacks led to the 23 dead transitions in the state-space report. The KCIA was used at the transport layer since the attacker already knew the pre-shared key used by the devices, enabling the attacker to decrypt MSG1. The attacker could access the session key after decrypting MSG1. The attacker could successfully acquire the random number and CMAC value throughout the subsequent communication procedure using the session key. Using the query statement, Figure 9 represents the secret information obtained by the attacker after launching a KCIA. Moreover, it can be found in the attacker’s places that the sender and the receiver used the same identity ID during the two communications, so the attacker could evaluate the user’s life based on the device identity information.

![Figure 8. Identity authentication result on the receiver.](image1)

![Figure 9. Attacker obtains information query results.](image2)
The EnOcean protocol state-space analysis revealed that the protocol could successfully resist replay attacks and tamper attacks. The EnOcean protocol, however, was unable to resist the KCIA. An attacker could successfully gain the session key used by the device during communication if the attacker broke the pre-shared key. As a result, the attacker had access to all the private information shared via the communication devices. This protocol did not ensure the anonymity of the communication devices since the sender and receiver’s identification did not change during the two communication operations.

5. Device Anonymity and Key Update Scheme Based on EnOcean Protocol

5.1. EnOcean-A Protocol

In order to address that the communication device was not anonymized in the EnOcean protocol and was insecure against a KCIA, this paper proposed the EnOcean-A protocol. Under the premise of low performance of chip and memory, we introduced a trusted third-party server to send communication keys to communication devices, and communication devices must obtain communication keys from the trusted third-party server every time they communicated. Moreover, the homomorphic hash function was used to anonymize the device identity information, and the dynamic update mechanism of the device identity information was introduced. In the EnOcean-A protocol, when the communication device started the teach-in process, first the sender initiated a communication key application request to the trusted third-party server, and the trusted third-party server, after verifying the identities of the sender and the receiver, sent the anonymous identity information and communication key of the receiver to the sender, sent the anonymous identity information and communication key of the sender to the receiver, and the trusted third-party server updated the hash of the device information. When the sender and the receiver received the identity information and communication key from the trusted third-party server, they first updated the identity information hash of their own device, and then the communicating parties started the teach-in process and authentication process based on the original EnOcean protocol.

5.2. EnOcean-A Protocol Communication Process

The specific communication process of the EnOcean-A protocol is shown below. Table 6 lists the important symbols, and Figure 10 represents the specific communication process of the EnOcean-A protocol.

Table 6. Symbolic representation of EnOcean-A communication process.

| Symbol | Definition                        |
|--------|----------------------------------|
| OldID<sub>X</sub> | Old ID of device X               |
| NewID<sub>X</sub> | New ID of device X               |
| H(,)   | One-way hash function            |
| r<sub>i</sub>, n<sub>i</sub> | <i>i</i>th random number        |
| Str<sub>i</sub> | String                          |
| Device<sub>i</sub> | Device name                      |
| RLC     | Rolling data                     |
| p       | Payload                          |
| SK      | Session key                      |
| CK      | Communication key                |
| RND     | Random number                    |
| HID<sub>i</sub> | Hash value of device <i>i</i>’s ID|
| A, B, S | Device A, Device B, and Server S  |
| CMAC(.) | Cipher-based message authentication code |
| ⊕      | XOR operation                     |
| ||     | Concatenation operation           |
| Enc(.) | Encryption function              |
Figure 10. EnOcean-A protocol data flow.

- **Teach-in phase:**

  Step 1: When $A$ sends a request for authentication to $S$, $A$ picks a random number $r_a$ and computes $a = H(r_a)$ and $x = H(\text{OldID}_A + r_a)$. $A$ sends $(a, x)$ to $S$.

  Step 2: After $S$ receives $(a, x)$, $S$ finds the stored user identification hash function group, calculates $x' = H(\text{OldID}_i) \times a$, and determines whether $x = x'$ exists. If it exists, then $A$ authentication is successful; otherwise, authentication is not passed. After
the authentication is passed, S selects a random number $r_b$ and calculates $\beta = H(r_b)$, \( y = H(\text{OldID}_A) \times \alpha \times \beta \). S generates two random numbers $n_1$ and $n_2 (0 < n_1, n_2 < L, n_1 \neq n_2)$, where $L$ is the length of $y$, and obtains three strings: $\text{Str}_1$, $\text{Str}_2$, and $\text{Str}_3$. $\text{Str}_1$ is the data $y$ from the start position to position $n_1$, $\text{Str}_2$ is the data $y$ from position $n_2$ to the end position, and $\text{Str}_3$ is the data $y$ from $n_1$ to position $n_2$. S sends ($\text{Str}_1$, $n_1$, $n_2$, $\beta$) to A after the calculation is finished. At the same time, S needs to update the hash value inside the function group: $H(\text{OldID}_A) = H(\text{NewID}_A)$, $H(\text{NewID}_A) = H(\text{NewID}_A) \times \alpha$.

Step 3: After A receives the message, $y' = H(\text{OldID}_A) \times \alpha \times \beta$ is calculated, $\text{Str}_1'$, $\text{Str}_2'$, and $\text{Str}_3'$ are calculated by $n_1$ and $n_2$ and $\text{Str}_1$ is compared with $\text{Str}_1'$. If equal, it means A considers S authenticated successfully; otherwise, authentication fails. After successful authentication, A calculates $z = \text{Device}_B \oplus \text{Str}_2'$ and A sends ($z$, $\text{Str}_2'$) to S. At the same time A updates the hash value of identity: $H(\text{OldID}_A) = H(\text{NewID}_A)$, $H(\text{NewID}_A) = H(\text{NewID}_A + r_b)$.

Step 4: After receiving the data sent from A, S first compares $\text{Str}_2'$ with $\text{Str}_2$. If $\text{Str}_2' = \text{Str}_2$, S considers A authenticated successfully and uses $\text{Str}_2$ to perform an XOR operation on the message $z$ to obtain $\text{Device}_B$ and learns that A wants to communicate with B. S generates a random number $r_2$ and calculates $\gamma = H(r_2), p = H(\text{OldID}_B) \times \gamma$. S generates two random numbers $n_3$ and $n_4 (0 < n_3, n_4 < L, n_3 \neq n_4)$, where $L$ is the length of $p$, to obtain $\text{Str}_4$, $\text{Str}_5$, and $\text{Str}_6$. $\text{Str}_4$ is the data $p$ from the start position to position $n_3$, $\text{Str}_5$ is the data $p$ from position $n_4$ to the end position, and $\text{Str}_6$ is the data from $n_3$ to position $n_4$. ($\text{Str}_4$, $n_3$, $n_4$, $\gamma$) is sent to B when the calculation is finished.

Step 5: After B receives the message ($\text{Str}_4$, $n_3$, $n_4$, $\gamma$), $y' = H(\text{OldID}_B) \times \gamma$ is computed, and $\text{Str}_4'$, $\text{Str}_5'$, and $\text{Str}_6'$ are compared. If equal, it means that B considers S authenticated successfully; otherwise, authentication fails. After successful authentication, B sends $\text{Str}_4'$ to S. At the same time, B updates the hash value of ID: $H(\text{OldID}_B) = H(\text{NewID}_B)$, $H(\text{NewID}_B) = H(\text{NewID}_B) \times \gamma$.

Step 6: After S receives the message, $\text{Str}_5$ is compared with $\text{Str}_5'$. If equal, then S considers B authenticated successfully; otherwise, authentication fails. After successful authentication, S generates the communication key $\text{CK}$, calculates $K_1 = (\text{CK} \parallel H(\text{ID}_A)) \oplus \text{Str}_6$ and $K_2 = (\text{CK} \parallel H(\text{ID}_B)) \oplus \text{Str}_5$, and finally sends $K_1$ to B and $K_2$ to A. At the same time, S needs to update the hash value: $H(\text{OldID}_B) = H(\text{NewID}_B)$, $H(\text{NewID}_B) = H(\text{NewID}_B)$ $\times \gamma$.

Step 7: After receiving the message, A uses $\text{Str}_3$ to perform an XOR operation on message $K_2$ to obtain the communication key $\text{CK}$ and $\text{HID}_B$. After receiving the message, B uses $\text{Str}_6$ to perform an XOR operation on message $K_1$ to obtain the communication key $\text{CK}$ and $\text{HID}_A$.

Step 8: A generates a rolling code $\text{RLC}$ and a session key $\text{SK}$ and then computes $M_1 = E(\text{CK}, (\text{RLC}, \text{SK}))$. A sends ($\text{HID}_A$, $\text{HID}_B$, $M_1$) to B.

Step 9: After receiving the message, B decrypts $M_1$ with the communication key $\text{CK}$ and obtains the rolling code $\text{RLC}$ and the session key $\text{SK}$. The teach-in phase ends.

- Authentication phase:

Step 10: A sends ($\text{HID}_A$, $\text{HID}_B$) to B.

Step 11: After B receives the message, B generates a random number $\text{RND}$ and sends ($\text{HID}_A$, $\text{HID}_B$, $\text{RND}$) to A.

Step 12: After the message is received by A, if the verification is successful, A calculates $M_2 = \text{CMAC}(\text{SK}, (p, \text{RND}))$ and $M_3 = E(\text{SK}, (p, M_2))$ and sends ($\text{HID}_A$, $\text{HID}_B$, $M_3$) to B.

Step 13: After B receives the message, B obtains the $p$ and CMAC using the session key $SK$. B calculates $M_4 = \text{CMAC}(SK, (p, RND))$, and if $M_4 = M_2$, the data are confirmed to be complete, and B considers A authenticated successfully. The specific process is shown in Figure 10.
6. Formal Analysis of EnOcean-A Protocol

6.1. EnOcean-A Protocol HCPN Model

The EnOcean-A protocol was modeled using CPN Tools. The EnOcean-A protocol’s top model had 28 places and 8 substitution transitions. The substitution transition Authentication_Server stood in for the process by which the sender made a communication key request to the trusted third-party server and acquired the communication key and the receiver’s identification hash value. The substitution transition Authentication_A stood in for the process where the trusted third-party server transmitted a communication key and the hash value of the receiver’s identity to the sender. The substitution transition Authentication_B represented the module that the trusted third-party server transmitted to the receiver for both the communication key and the hash value of the sender’s identity. The substitution transition Authentication_Server’ represented the process through which the receiver acquired the communication key and the hash value of the sender’s identity. The substitution transition Send_Teach-in represented when the sender constructed the teach-in message. The substitution transition Receive_Teach-in represented that the receiver decrypted teach-in messages from the sender. The substitution transition Send_Message represented that the sender submitted the authentication request. The substitution transition Send_Message’ represented the process of verifying the sender for the receiver. Figure 11 depicts the specific procedure.

Figure 11. EnOcean-A top-level model.

Figure 12 shows the detailed process for the substitution transition of Authentication_Server. There were four steps in this transition, which were: the sender sent the communication key request to the server, the sender received the authentication request from the server, the sender saved the communication key and hash value of the receiver’s identity from the server, and the sender updated the hash value of its identity.
The specific process of sending a communication key request from the sender to the server is shown below. The place SelectRandom sent a random number to the transition Compute_Hash1, and the transition Compute_Hash1 calculated the hash of this random number, sent the calculated hash to the transition SendToServer, and sent the random number to the transition Compute_Hash2. The hash' was calculated using the random number and the identity ID, and the hash' was sent to the transition SendToServer. Finally, the transition SendToServer sent the two hashes to the place Send_MSG1.

The specific process of updating the identity ID at the sending end was as follows. When MSG1 was sent successfully, the transition Update_OldID obtained the new identity ID of the device and sent the new identity ID to the transition Update_NewID. The transition Update_NewID saved the received identity ID to the place OldID' and used the received identity ID and a random number to calculate the new identity ID of the device, saving the new identity ID to the place NewID'.

The specific process of receiving an identity request from the server at the sending end was as follows. Place Rec_MSG2 sent the message to the transition Split_MSG2. The transition Split_MSG2 decomposed the received message, sent str to the transition Compare_Str, and sent hash', random', and random to the transition Compute_Str. The transition Compute_Hash1' used the previously calculated hash'' and the received hash' to calculate the new hash and sent the hash to the transition Compute_Hash2. The transition Compute_Str used the random numbers random' and random to decompose the hash into str, str', and str'' and saved str to the place str1'. str'' was saved to place str2', and str'' was saved to place str3'. The transition Compare_Str compared str'' with str. If different, the subsequent steps were stopped. If not, str'' was sent to the transition Compute_XOR, which calculated the difference between id and XOR and sent it to the transition Send_MSG3.
The process of saving the communication key from the server and the receiver’s identity hash on the sending side was as follows. Place Rec_MSG7 sent the received message to the transition XOR, which used str′ to compute the heterogeneous value of the received message and sent the computation structure to the place Store_Content1.

Figure 13 shows the internal model of substitution Authentication_A. This substitution transition consisted of three specific processes: the server authenticated the sender, the server sent the authentication request to the sender, the server sent the authentication request to the receiver, and the server sent the communication key and the receiver’s identity hash value to the receiver.

The specific process of the authentication of the sender by the server was as follows. The place Rec_MSG1 sent the message to the transition Compare_Hash, which compared the hash in the message to prevent replay attacks. If a replay attack was detected, all the subsequent steps were terminated. Otherwise, the message was sent to the transition Split_MSG1. The transition Split_MSG1 split the message, sent the hash to the transition ComputeHash, and sent the hash′ to the transition CompareHash. The transition ComputeHash calculated the hash′ from the hash′. If hash_x was the same, the server authenticated the sender successfully, and the transition Update_ID′ and the transition Update_ID updated the hash value of the sender, saving the result to place DBHashID.

The process of sending an authentication request from the server to the sender was as follows. The transition Compute_Hash calculated the hash of a random number and sent the calculated hash to the transition Compute_Hash1. The transition Compute_Hash1 calculated the hash, hasholda, and hash′ and sent the hash to the transition Separate_Y. The
transition Separate_Y divided the incoming hash into str, str', and str'' strings. Finally, the transition Send_MSG2 sent str, two random numbers, and hash' to the place Send_MSG2.

The exact process of sending an authentication request from the server to the receiver is shown below. Place Rec_MSG3 sent the message to the transition Split_MSG3. The transition Split_MSG3 split the received message and sent str to the transition Compare_str2. The transition Compare_str2 compared the received str with the str' calculated by the server, and if it was the same, it meant the server successfully authenticated the receiver. The server sent the str to the XOR, which computed the message (id, str) and sent the result to the place store_ID.

The server sent the communication key and the receiver’s identity hash to the receiver as shown below. Place SendIDb sent the received message to the transition Concatenate, which sent the receiver’s identity hash and communication key to the transition XOR'. The transition XOR' performed an XOR operation between the received message and the string str and sent the result to the place Send_MSG7.

6.2. EnOcean-A Protocol Security Assessment

The Dolev–Yao attacker model was introduced to attack the network level of the new scheme, including a tamper attack, replay attack, and KCIA. The purple part indicates the replay attack, the red part indicates the tamper attack, and the blue part indicates the KCIA. The details are shown in Figures 14 and 15.

Figure 14. Tamper attack and replay attack at the network transport layer.
Figure 15. KCIA at the network transport layer.

Table 7 compares the state-space report of the EnOcean-A protocol with the state-space report of the EnOcean-A protocol after adding the attack.

Table 7. State-space analysis of EnOcean-A protocol.

| Type                | EnOcean-A | Tamper Attack | Replay Attack | KCIA  |
|---------------------|-----------|---------------|---------------|-------|
| State-Space Nodes   | 726       | 48            | 9             | 1326  |
| State-Space Arcs    | 944       | 47            | 8             | 2224  |
| Scc Graph Nodes     | 726       | 48            | 9             | 1326  |
| Scc Graph Arcs      | 944       | 47            | 8             | 2224  |
| Dead Markings       | 1         | 1             | 1             | 1     |
| Dead Transitions    | 0         | 40            | 81            | 4     |

According to Table 7, the EnOcean-A protocol’s state space report shows that there was one dead node and zero dead transitions, indicating that the protocol could operate normally and that identity authentication between the sender and the receiver was successful. The EnOcean-A protocol added the tamper attack. A communication could be tampered with because the receiver delivered the authentication result in plain text. The 40 dead transitions were the result of the failed authentication, which the server detected when it checked the message and discovered that it did not match. The protocol added the replay attack since the sender created the information request while requesting the communication key from the server using a random number. The server assessed the message’s random number after receiving the sender’s communication key request for information. When the random number was found to be received, the key request failed due to instant determination that a replay attack occurred. The server identified a replay attack that resulted in 81 dead transitions. The KCIA was introduced in the protocol because the sender requested the communication key from the server before communicating with the receiver, so the communication key used by the communicating parties was different each time they communicated, and the four dead changes were due to the attacker’s inability to break the encrypted message. By querying the repository of the attacker’s intercepted content, it can be found that, since the hash value of the identity information was updated
after the receiver and the recipient finish communicating with the server, a different hash value was used for each communication, so the attacker could not obtain the specific information of the communicating device, as shown in Figure 16. By querying the server’s place, it was found that the trusted third-party server updated the hash of the device’s identity information at each communication, as shown in Figure 17. Since the EnOcean-A protocol updated the communication key at each communication, the attacker could not decrypt the obtained information and could not know the detailed content of the message transmission. Thus, the attacker could not launch a tamper attack, replay attack, or KCIA on the EnOcean-A protocol. Moreover, the EnOcean protocol used a homomorphic hash function to anonymize the device identity, so the attacker could not determine whether a message was sent from a device.

![Figure 16. Information search in attacker’s place.](image1)

![Figure 17. Information search in trusted third-party server’s place.](image2)

6.3. EnOcean-A Security Analysis

In the following, the proposed scheme was secure against various known attacks.

- **Replay attack**: The attacker eavesdropped the transmitted data and resent the eavesdropped data to the receiver in the next round of inter-node communication to achieve the purpose of cheating the receiver. In the EnOcean-A protocol, the sender and receiver sent hash values that generated random numbers when communicating with the server, so when the device received a message, it first determined whether the random number hash value in the message already existed; if it did, it directly discarded the message. The teach-in message sent by the sender also included a random number. When the receiver received the message, it checked whether the random number was already received. If it was found that the random number was already received, then the message was discarded directly.

- **Impersonation attack**: The attacker used the intercepted device identity information to initiate a session request to other devices. In the EnOcean protocol, the attacker could not obtain the device’s identity launch impersonation attack because the device information was anonymized before communication, and the device anonymization information was updated at each communication.

- **Tamper attack**: The attacker intercepted the message transmitted using plaintext and sent the tampered message to the receiving end. In the EnOcean-A protocol, the sender and receiver used partial hashes of identity for transmission when communicating with the server. Even if an attacker intercepted and tampered with the hash value of the message, when the two communicating parties received the message, they compared it with the hash value of their own saved identity and discarded the message if they found that the hash value was different. In the authentication phase, the sender used the session key to encrypt the calculated CMAC value, and the attacker could not tamper with the message without knowing the session key.
• Eavesdropping attack: The attacker used a passive attack to eavesdrop on the data transmitted in the network and analyzed the data to launch an attack on the node. In the EnOcean-A protocol, since the sender and receiver sent partial hash values of the devices when communicating with the server, even if the attacker eavesdropped on the transmitted messages, they could not construct a complete hash of the device identity.
• KCIA: In the EnOcean-A protocol, the communication key was sent from the trusted third-party server to both communicating parties, and both communicating parties obtained the communication key from the trusted third-party server before each communication. Even if an attacker broke the communication key used during a certain communication, he could not use this communication key to decrypt the contents of messages transmitted before or after. The freshness of the communication key was also guaranteed. This scheme could provide backward and forward security.
• Anonymity: In the EnOcean-A protocol, the sender must obtain the hash of the receiver’s identity from the trusted third-party before communicating with the receiver, and when the sender communicated with the receiver, the sender also used the hash of the identity for message delivery. Therefore, the communication process could ensure the anonymity of device information.
• Unlinkability: In the EnOcean-A protocol, the sender used an updated hash each time it communicated with the trusted third-party, and the trusted third-party server also updated the hash of the receiver. Therefore, the attacker could not deduce the specific communication device from all the hash values eavesdropped after obtaining the communication information.

Due to the limited memory and performance of EnOcean device chips, the new protocol introduced a trusted third-party server to address the EnOcean protocol’s security issues. To guarantee that the protocol was resistant to a KCIA, the trusted third-party server produced and delivered the communication key to the communication parties. The EnOcean-A protocol used a homomorphic hash function that could be run securely on lightweight devices such as EnOcean. In addition, the EnOcean-A protocol could ensure that communication devices remained untraceable and anonymous. Table 8 compared the security analysis of EnOcean-A protocol with EnOcean protocol. The data in the table fully showed that EnOcean-A protocol has stronger security.

| Protocol  | Tamper Attack | Replay Attack | Impersonation Attack | KCIA | Anonymity | Unlinkability |
|-----------|---------------|---------------|----------------------|------|-----------|---------------|
| EnOcean   | ×             | ×             | ✓                    | ✓    | ✓         | ✓             |
| EnOcean-A | ×             | ×             | ×                    | ×    | ×         | ×             |

7. Conclusions

This paper addressed the security problem that sensitive data transmitted by devices in intelligent lighting control systems can be easily obtained by malicious attackers and used to analyze the personal privacy of users based on intercepted messages. It took the EnOcean protocol as the research object. The formal analysis of the EnOcean protocol using CPN theory and the Dolev–Yao attacker model revealed that the EnOcean protocol was insecure against a KCIA, and the identity information of communication devices was not anonymized, so an attacker could analyze a user’s life status based on the eavesdropped device identity information. To address the above issues and due to the limited computing power and storage control of EnOcean devices, a EnOcean-A protocol was proposed. The improved protocol introduced a trusted third-party server to send communication keys, as well as each time the communication device needed to obtain a new communication key from the trusted third-party server, which was designed to resist a KCIA and provide forward security. The device identity information was dynamically updated so that the attacker could not analyze the user’s usage from the eavesdropped device identity information, ensuring the unlinkability of the device identity information. A formal analysis of
the EnOcean-A protocol revealed that the improved protocol could effectively resist the security problems existing in the EnOcean protocol and provide stronger security. In this paper, we only focused on improving the security of the protocol and did not sufficiently consider the real-time aspect of the protocol communication. In future work, we plan to consider the protocol to improve the security and reduce the time cost to achieve real-time requirements as much as possible.

**Author Contributions:** T.F. performed the data collection and review. Y.W. conducted the software implementation and wrote the initial draft. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by the National Natural Science Foundation of China (grant number 61762060) and the Foundation for the Key Research and Development Program of Gansu Province, China (grant number 20YF3GA016).

**Data Availability Statement:** No data were used to support this study.

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

**References**

1. Soheilian, M.; Fischl, G.; Aries, M. Smart Lighting Application for Energy Saving and User Well-Being in the Residential Environment. *Sustainability* 2021, 13, 6198. [CrossRef]
2. Shen, S. Practice and Research on the Combination of Interior Design and Smart Home in Internet Information Age. In Proceedings of the 2021 International Conference on Computer Technology and Media Convergence Design (CTMCD), Sanya, China, 23–25 April 2021; pp. 273–276.
3. Schomakers, E.M.; Hannah, B.; Martina, Z. Users’ Preferences for Smart Home Automation—Investigating Aspects of Privacy and Trust. *Telemat. Inform.* 2021, 64, 101689. [CrossRef]
4. Iqbal, W.; Haider, A.; Bilal, R.; Yawar, A.; Faisal, A.; Ahmed, H. PCSS: Privacy Preserving Communication Scheme for SDN Enabled Smart Homes. *IEEE Sens. J.* 2021, 1, 7779. [CrossRef]
5. Ompal; Vishnu, M.M.; Adesh, K. FPGA Integrated IEEE 802.15.4 ZigBee Wireless Sensor Nodes Performance for Industrial Plant Monitoring and Automation. *Nucl. Eng. Technol.* 2022, 54, 2444–2452. [CrossRef]
6. Farha, F.; Ning, H. Enhanced Timestamp Scheme for Mitigating Replay Attacks in Secure ZigBee Networks. In Proceedings of the 2019 IEEE International Conference on Smart Internet of Things (SmartIoT), Tianjin, China, 9–11 August 2019; pp. 469–473. [CrossRef]
7. Khanji, S.; Farkhund, I.; Patrick, H. ZigBee Security Vulnerabilities: Exploration and Evaluating. In Proceedings of the 2019 10th International Conference on Information and Communication Systems (ICICS), Irbid, Jordan, 11–13 June 2019; pp. 52–57.
8. Chen, X.Y.; Chen, Y.X.; Yu, Q.M. Smart Home System with Bluetooth and Wi-Fi as Communication Mode. In Proceedings of the 2021 International Conference on Digital Society and Intelligent Systems (DSiSnS), Chengdu, China, 19–21 November 2021; pp. 143–147.
9. Mori, G.N.; Swaminarayan, P.R. Measuring IoT Security Issues and Control Home Lighting System by Android Application Using Arduino Uno and HC-05 Bluetooth Module. In *Data Science and Intelligent Applications*; Springer: Singapore, 2021; pp. 375–382.
10. Lonzetta, A.M.; Cope, P.; Joseph, C.; Bassam, J.M.; Thaier, H. Security Vulnerabilities in Bluetooth Technology as Used in IoT. *J. Sens. Actuator Netw.* 2018, 7, 28. [CrossRef]
11. Zhang, Y.; Jian, W.; Rajib, D.; Jin, Y.E.; Lin, Z.Q.; Fu, X.W. On the (In)Security of Bluetooth Low Energy One-Way Secure Connections Only Mode. *arXiv* 2019, 1908, 10497.
12. Arshad, J.; Aziz, M.; Al-Huqail, A.A.; Zaman, M.H.U.; Husnain, M.; Rohman, A.U.; Shafiq, M. Implementation of a LoRaWAN Based Smart Agriculture Decision Support System for Optimum Crop Yield. *Sustainability* 2022, 14, 827. [CrossRef]
13. Hofer, F.; Barbara, R. Architecture and Its Vulnerabilities in Smart-Lighting Systems. In *Technologies for Smart Cities*; Springer: Boston, MA, USA, 2021; pp. 155–181. [CrossRef]
14. Kannayeram, G.; Madhumitha, M.; Mahalakshmi, S.; Menaga Devi, P.; Monika, K.; Prakash, N.B. Smart Environmental Monitoring Using LoRaWAN. In Proceedings of the 3rd International Conference on Communication, Devices and Computing, Haldia, India, 16–18 August 2022; pp. 513–520. [CrossRef]
15. Hakeem, S.A.A.; El-Kader, S.M.A.; Kim, H. A Key Management Protocol Based on the Hash Chain Key Generation for Securing LoRaWAN Networks. *Sensors* 2021, 21, 5838. [CrossRef]
16. Han, J.; Wang, J. An Enhanced Key Management Scheme for LoRaWAN. *Cryptography* 2018, 2, 34. [CrossRef]
17. You, I.; Kwon, S.; Choudhary, G.; Sharma, V.; Seo, J.T. An Enhanced LoRaWAN Security Protocol for Privacy Preservation in IoT with a Case Study on a Smart Factory-Enabled Parking System. *Sensors* 2018, 18, 1888. [CrossRef]
18. Naoui, S.; Mohamed Elhoucine, E.; Leila, A.S. Novel Enhanced LoRaWAN Framework for Smart Home Remote Control Security. *Wirel. Pers. Commun.* 2020, 110, 2109–2130. [CrossRef]
19. Sanchez-Iborra, R.; Sánchez-Gómez, J.; Pérez, S.; Fernández, P.J.; Santa, J.; Hernández-Ramos, J.L.; Skarmeta, A.F. Enhancing LoRaWAN Security through a Lightweight and Authenticated Key Management Approach. *Sensors* 2018, 18, 1833. [CrossRef] [PubMed]

20. Butun, I.; Nuno, P.; Mikael, G. Analysis of LoRaWAN v1.1 Security: Research Paper. In Proceedings of the 4th ACM MobiHoc Workshop on Experiences with the Design and Implementation of Smart Objects, New York, NY, USA, 25 June 2018. [CrossRef]

21. Tsai, K.L.; Fang-Yie, L.; Li-Ling, H.; Chia-Yin, K. Secure Session Key Generation Method for LoRaWAN Servers. *IEEE Access* 2020, 8, 54631–54640. [CrossRef]

22. ISO/IEC 14543-3-10:2012. Available online: http://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/05/98/59865.html (accessed on 6 August 2022).

23. Kambourakis, G.; Kolias, C.; Genetatakis, D.; Karopoulos, G.; Makrakis, G.M.; Kounelis, I. A State-of-the-Art Review on the Security of Mainstream IoT Wireless PAN Protocol Stacks. *Symmetry* 2020, 12, 579. [CrossRef]

24. Sadri, M.J.; Asaar, M.R. An Anonymous Two-Factor Authentication Protocol for IoT-Based Applications. *Comput. Netw.* 2021, 199, 108460. [CrossRef]

25. Banerjee, S.; Odelu, O.; Das, A.K.; Jangirala, S.; Kumar, N.; Chattopadhyay, S.; Choo, K.K.R. A Provably-Secure and Lightweight Anonymous User Authenticated Session Key Exchange Scheme for Internet of Things Deployment. *IEEE Internet Things J.* 2019, 6, 8739–8752. [CrossRef]

26. Fakroon, M.; Alshahrani, M.; Gebali, F.; Traore, I. Secure Remote Anonymous User Authentication Scheme for Smart Home Environment. *Internet Things* 2020, 9, 100158. [CrossRef]

27. Hajian, R.; Haghhighat, A.; Erfani, S.H. A Secure Anonymous D2D Mutual Authentication and Key Agreement Protocol for IoT. *Internet Things* 2022, 18, 100493. [CrossRef]

28. Li, F.Y.; Yu, X.Y.; Yang, C.; Yu, S.Q.; Sun, Y.H.; Wang, Y.L.; Zhou, H.Y. An Anonymous Authentication and Key Agreement Protocol in Smart Living. * Comput. Commun.* 2022, 199, 108460. [CrossRef]

29. Banerjee, S.; Odelu, V.; Das, A.K.; Chattopadhyay, S.; Park, Y. An Efficient, Anonymous and Robust Authentication Scheme for Smart Home Environments. *Sensors* 2020, 20, 1215. [CrossRef]

30. Hofer-Schmitz, K. A Formal Analysis of EnOcean’s Teach-in and Authentication. In Proceedings of the 16th International Conference on Availability, Reliability and Security, Vienna, Austria, 17–20 August 2021; pp. 1–8. [CrossRef]