*electronics*

MDPI

*Article*

# A Multiple End-Devices Authentication Scheme for LoRaWAN

**Chun-I Fan, Er-Shuo Zhuang, Arijit Karati * and Chun-Hui Su**

Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung 804, Taiwan; cifan@mail.cse.nsysu.edu.tw (C.-I.F.); zhuanges@gmail.com (E.-S.Z.); chunhye85315@gmail.com (C.-H.S.)
* Correspondence: arijit.karati@mail.cse.nsysu.edu.tw

**Abstract:** With the advancement of the Internet of Things, the LoRa Alliance produced the Long-Range Wide-Area Network (LoRaWAN) Specification, allowing end-devices to transit through a gateway and join the LoRa network after completing a join procedure. When an end-device joins the LoRaWAN network, it must send a join request message to the network server and wait for the network server to verify such request under the current LoRaWAN join protocol. However, as the number of end-devices grows exponentially, network server verification messages will grow linearly with the number of end-devices. This paper proposes an authentication system for multiple end-devices that complies with the LoRa Alliance's specifications and decreases the joining latency imposed by the network server verifying messages. The proposed authentication system is formally secure against the server and end-device impersonation. In addition, we assess the authentication overhead and compare it to the standard approach.

**Keywords:** authentication; Internet of Things (IoT); LoRaWAN; join procedure



**Citation:** Fan, C.-I.; Zhuang, E.-S.; Karati, A.; Su, C.-H. A Multiple End-Devices Authentication Scheme for LoRaWAN. *Electronics* **2022**, *11*, 797. https://doi.org/10.3390/electronics11050797

Academic Editor: Rui L. Aguiar

Received: 20 January 2022
Accepted: 1 March 2022
Published: 3 March 2022



## 1. Introduction

The IoT has enabled various applications in the past few years, including smart homes, smart cities, industrial automation, and smart healthcare [1,2]. According to Ericsson's mobility report [3], the number of IoT devices exceeded the number of mobile devices in 2018. Furthermore, as illustrated in Figure 1, according to a Statistica report in 2018 [4], IoT devices will have over 75 billion connections by 2025. Furthermore, since technology advances and times change swiftly, consumers have exhibited a greater demand for automated IoT devices and consume less power. This means that previous wireless communication technologies such as Bluetooth Low Energy (BLE), Zigbee, and Wi-Fi are no longer capable of meeting the needs of IoT devices for long distances, low energy consumption, and a high density of nodes. Furthermore, according to the LoRa Alliance [5] and Sinha et al. [6], the Low-Power Wide-Area Network (LPWAN) industry is on the rise in the IoT market. In this scenario, low-power wide-area network (LPWAN) communication technology has recently become a prominent wireless communication technology [7,8].

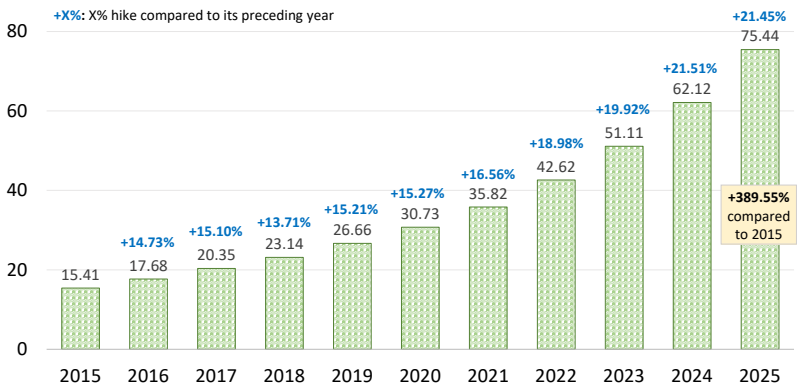


**Figure 1.** Connected IoT devices (in billions) from 2015 to 2025.

LoRa [9] is a well-developed technology among these LPWAN technologies. The LoRa technology was developed by the French company Cycleo and bought by Semtech in 2012. Following the acquisition, Semtech commenced an aggressive increase in the use of the technology, including forming the LoRa Alliance to make it easier for other firms to join the LoRa ecosystem. LoRa technology [10] bridges the technical gap between Wi-Fi/BLE and cellular-based networks, such as the 5G network [11], and can provide long-distance data connections with minimal power consumption, as shown in Figure 2. Cisco, one of the LoRa Alliance's pioneers, presented the following three LoRa application use cases [12]:

- Smart city management: Connecting all the assets of the city and providing better services by collecting data and analyzing data, including waste management, parking, street lighting, and public safety data.
- Asset tracking: Obtaining location information for people and private assets, including equipment, vehicles, pets, and cattle.
- Gas and water metering: Enabling utility customers to remotely read and control gas and water meters to reduce technical costs.

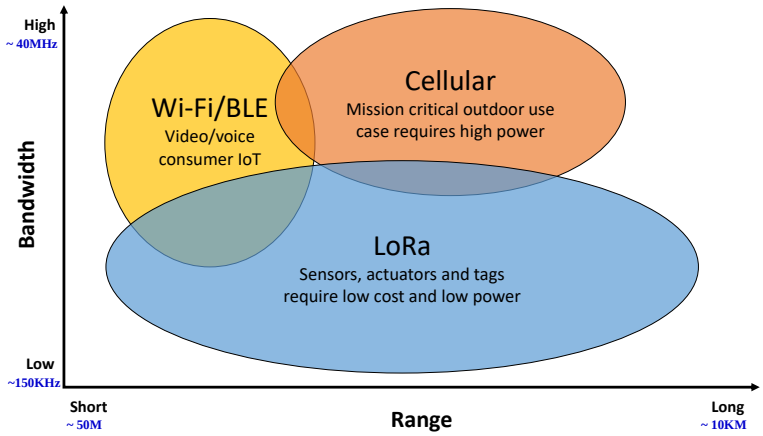


**Figure 2.** LoRa filling the technology gap.

The LoRa Alliance released the first version of the LoRa communication standard specifications in 2015, designating the Long Range Wide-Area Network (LoRaWAN) to enable LoRa to be applied to the abovementioned areas. According to the LoRa Alliance's introduction overview document issued in 2015 [13], LoRaWAN has the following benefits:

- Long range: Not only can LoRaWAN serve indoor applications in unlicensed bands such as Wi-Fi, but it can also support outdoor applications with the same security as a cellular network [14]. As a result, it can connect to a broader range of networks than Wi-Fi and cellular networks.
- Max lifetime: Asynchronous communication is the most extensively utilized mode for end-devices in the LoRaWAN network. End-devices only communicate over the network when sending event-driven or planned data. As a result, they can conserve the wake-up monitoring power. End-devices in other synchronous networks must wake up frequently in order to synchronize with the network and verify messages, which consumes more power.
- Multi-usage: Depending on the environment, the LoRaWAN network may handle public networks with a significant capacity of hundreds or even tens of thousands of nodes. As a result, it can meet the needs of large online communities or markets [15].
- Low cost: Low power consumption in LoRa end devices enhances battery life and lowers battery replacement expenses. Semtech has also made the hardware and software designs for the gateway and end-devices public. This can help to save money on infrastructure and terminal device development.
- Device classes: End-devices are used for a variety of purposes and have varying needs. LoRaWAN has created three end-device categories, dubbed class A, B, and C, to opti-

mize diverse end-application profiles to balance network downlink communication delays and battery life.
- Security: To ensure end-to-end security, LoRaWAN employs Advanced Encryption Standards (AES) [13]. Mutual authentication and integrity protection also ensure the integrity of LoRa networks, devices, and data [14].

Chen et al. [16] developed a key generation strategy for LoRaWAN that meets the security key randomization criterion. Danish et al. [17] developed a lightweight two-factor authentication system for the LoRaWAN join procedure based on blockchain technology, ensuring confidence between network servers. Later, Tsai et al. [18] established a session key generation mechanism that allows two servers to interact with each other, considerably improving the LoRaWAN Specification's undefined application-layer communication security securely. For LoRaWAN networks, Jabbari and Mohasefi [19] created a novel secure user-authenticated key establishment mechanism. It allows participants to verify each other's identities. Furthermore, it enables users and end-devices to establish a secure session key between themselves without trusting the network server unconditionally. Further, Kaven et al. [20] proposed a scheme to combat additional Sybil and man-in-the-middle threats. However, most of the methods do not consider the authentication of multiple end-devices and thus suffer from high latency during real-life deployment.

### *1.1. Contributions*

When an end-device wants to join a network, it must send a join request to the network server, which will execute the join verification process [21]. With the growing number of end-devices, the number of join verification tasks will rise linearly [13]. As a result, if many end-devices want to join the network, the network server will experience a delay when processing such large requests. We propose a multi-device authentication technique based on the LoRaWAN join mechanism to solve this challenge. Within the LoRaWAN framework, we make the following contributions:

- We offer a comprehensive authentication strategy based on the linked end-device(s) features that use exclusive-OR operations to achieve batch authentication.
- The suggested technique allows several end-devices of the same gateway to generate group keys. Supporting group key creation may be helpful for future applications because end-devices from the same gateway may have specific correlations. Furthermore, each end-device requires one additional hash operation.
- Under the formal model, the new technique is protected against major threats such as phoney end-devices, network servers, and the leakage of session keys.
- Considering multi-end-device authentication, the suggested technique saves overall execution time by around 33% compared to the LoRA Specification.

It may be noted that the security evaluation of the proposed scheme is based on the conventional IND-CCA symmetric encryption assumptions and pseudorandom permutation indistinguishability (PRP).

### *1.2. Organization*

The rest of this manuscript is organized as follows: Section 2 mentions some background knowledge relevant to the proposed scheme, including the LoRaWAN standard. The proposed authentication scheme based on the LoRaWAN standard is described in Section 3. The security models and proofs of the proposed scheme are provided in Section 4. Comparisons and concluding remarks are presented in Sections 5 and 6, respectively.

## 2. Preliminaries

This section provides background information for the proposed work, including the LoRaWAN architecture with its specifications, and the underlying security games.

### *2.1. The LoRaWAN Architecture*

LoRaWAN is a wide-area network that offers a bi-directional communication protocol for end-devices with long-range, low-power, and low-cost requirements [22]. Figure 3 depicts an overview of the LoRaWAN architecture. The entities defined in the LoRaWAN architecture are described below [13]:

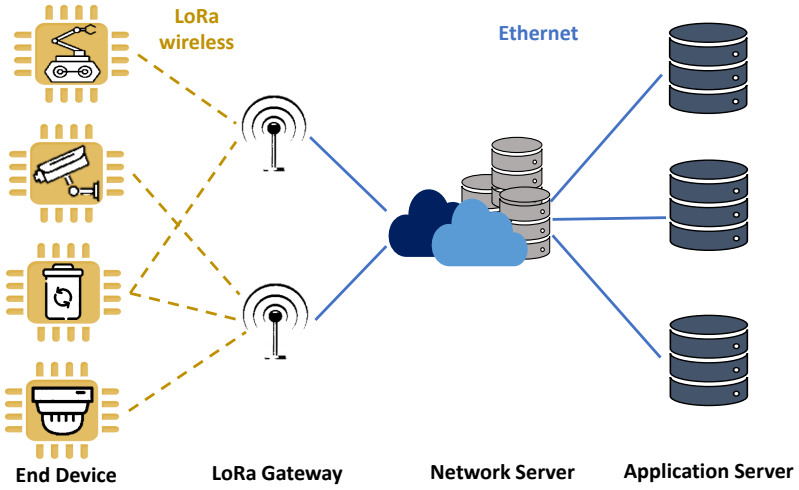


**Figure 3.** The LoRaWAN architecture.

- **End-Devices**: These could be anything from water meters to smoke alarms to trash cans and pet trackers. LoRaWAN divides end-devices into classes A, B, and C. Each type of device fulfills various long-distance communication requirements.
  - Class A: This is the most basic communication mode, with minor power consumption. It transmits data in bursts. As a result, network servers and applications cannot predict the communication time. Class A refers to end-devices that only require upload link communications.
  - Class B: Class B end-devices will open additional receive windows at predetermined times in addition to the random receive windows of class A devices. The gateway will send a time synchronization signal to the end-devices in order for them to open the receive windows on a predetermined schedule. As a result, the network server knows when the end-devices are listening.
  - Class C: The receive windows on class C end-devices are almost always open and only close when transmitting. This end-device consumes the most power, while having the shortest delay because these receive windows remain open.
- **Gateways**: A device in the LoRaWAN network is not assigned to a specific gateway. Instead, data sent by a node are typically received by multiple gateways. Each gateway will route data packets from the end node to the cloud-based network server via backhaul cellular or Ethernet networks.
- **Network Server**: This is in charge of managing the entire network through specific functions. It deletes redundant packets and runs security checks when it receives them from gateways. Finally, it sends back an accept message via the best gateway.
- **Application Servers**: This is in charge of interpreting data from end-devices and solving problems with advanced technologies such as deep learning or artificial intelligence.

### *2.2. The LoRaWAN Specifications*

When the end-device connects to the LoRaWAN network, the following data are stored in the end-device [23]:

- Device EUI (*DevEUI*): The *DevEUI* is a global unique end-device identification number that is assigned by the device manufacturer.

- Application EUI (*AppEUI*): The *AppEUI* is a global unique application identification number, which is stored in the device in advance by the device manufacturer.
- Application key (*AppKey*): The *AppKey* is an AES-128 secret key stored in the end-device and used to derive two keys, *NwkSKey* and *AppSKey*.
- Device address (*DevAddr*): The *DevAddr* is a 32-bit hexadecimal number, which can be divided into two parts: the network identifier and network address.
  - Network identifier (*NwkID*): The *NwkID* is a 7-bit network identifier used to distinguish between the addresses of overlapping networks operated by different network operators.
  - Network address (*NwkAddr*): The *NwkAddr* is a 25-bit network address assigned by the network manager.
- Network session key (*NwkSKey*): The *NwkSKey* is a network session key which is used for MAC layer message encryption and authentication.
- Application session key (*AppSKey*) The *AppSKey* is an application session key which is used for application-layer message encryption and authentication.

There are two ways for end-devices to join LoRaWAN, according to current LoRaWAN standards. The first is known as over-the-air activation (OTAA), and the second is known as activation by personalization (ABP). We will introduce them briefly.

#### 2.2.1. End-Device Activation

When using OTAA, the end-device must complete the join procedure to connect to the LoRaWAN network. *DevAddr*, *NwkSKey*, and *AppSKey*, on the other hand, are stored directly in the end-device when using ABP to join LoRaWAN. Because the end-device stores the information required for activation, it can join a specific LoRa network when it boots up. The proposed scheme focuses on the join procedure in OTAA mode.

#### 2.2.2. Join Procedure

The join procedure according to LoRaWAN Specification 1.0.3 is depicted in Figure 4. A join-request message and a join-accept message are sent during the join procedure. Before the end-device begins the join procedure, it must be personalized with the following data: *DevEUI*, *AppEUI*, and *AppKey*. The following is a description of the join procedure. First, the end-device sends a join-request message to the network server, including the *DevEUI*, *AppEUI*, device nonce (*DevNonce*), and message integrity code (MIC), which is computed using the *AppKey*. After verifying the join-request message, the network server derives the *NwkSKey* and *AppSKey* as follows:

$$NwkSKey = E_{AppKey}(\text{0x01}||AppNonce||NetID||DevNonce||pad_{16}) \quad (1)$$

$$AppSKey = E_{AppKey}(\text{0x02}||AppNonce||NetID||DevNonce||pad_{16}) \quad (2)$$

where 0x01 and 0x02 are the port fields for specific applications, *AppNonce* is an application nonce, *NetID* is a network identifier, and $pad_{16}$ is a function adding zero octets to make the length of the data a multiple of sixteen. The network server then sends *AppSKey* to the application server over a secure channel and a join-accept message to the end-device. The join-accept message contains *AppNonce*, *NetID*, *DevAddr*, a delay (*RxDelay*) between TX and RX, a reserved-for-future-usage field (*RFU*), and an optional list of channel frequencies for the network (*CFList*). *AppKey* encrypts the join-accept message as follows:

$$join\text{-}accept = E_{AppKey}(AppNonce||NetID||DevAddr||RFU||RxDelay||CFList||MIC) \quad (3)$$

where *MIC* is computed using *AppKey*. On receiving the join-accept message, the end-device computes *NwkSKey* and *AppSKey* as follows:

$$NwkSKey = E_{AppKey}(\text{0x01}||AppNonce||NetID||DevNonce||pad_{16}) \quad (4)$$

$$AppSKey = E_{AppKey}(\text{0x02}||AppNonce||NetID||DevNonce||pad_{16}) \quad (5)$$

where *AppNonce* is an application nonce and *NetID* is a network identifier.

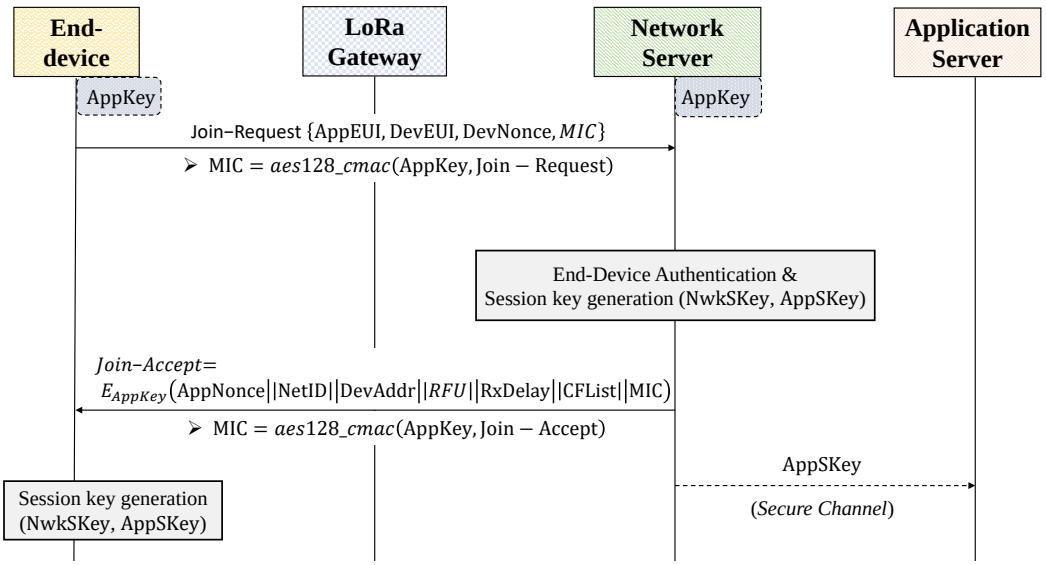


**Figure 4.** Join procedure in the LoRaWAN specification.

## 3. The Proposed Scheme

Table 1 displays the useful notations. This section describes an efficient authentication scheme for massive end-devices in the LoRaWAN join procedure. It is divided into three stages: *setup*, *personalization*, and *authentication*, as shown in Figure 5.

**Table 1.** A list of notations.

| Notation | Meaning |
|---|---|
| $AppEUI$ | Application global unique identifier |
| $DevEUI_i$ | Global unique identifier of the end-device $i$ |
| $AppKey_i$ | Secret key for the end-device $i$ |
| $DevNonce_i$ | Nonce generated by the end-device $i$ |
| $MIC$ | Message integrity code |
| $aes128\_cmac(K, M)$ | MAC function based on the AES-128 for secret key $K$ and message $M$ |
| $MHDR$ | MAC header |
| $r_i$ | Random number for the end-device $i$ |
| $c$ | Common random number |
| $NetID$ | Network identifier |
| $RFU$ | Reserved field for the future use |
| $RxDelay_i$ | The delay field of the end-device $i$ |
| $CFList$ | Optional list of channel frequencies |
| $pad_{16}$ | Zero octets for padding such that data length is a multiple of 16 |
| $H$ | Hash function $H : \{0,1\}^* \rightarrow \{0,1\}^\lambda$ |
| $AppSKey_i$ | Application session key for the end-device $i$ |
| $NwkSKey_i$ | Network session key for the end-device $i$ |
| $CK$ | Common session key |

### 3.1. Setup

The network server sends an authentication slot to the gateway. Then, the manufacturer chooses a one-way hash function and embeds it to end-devices. For each end-device $_i$, it shares with each network server the identity $DevEUI_i$ of the end-device $_i$ and a secret key $AppKey_i$ shared with the end-device $_i$. It then publishes the details of symmetric encryption and decryption, and one-way hash functions. Moreover, the device manufacturer sets the parameters according to the LoRaWAN Specifications of the LoRa Alliance.

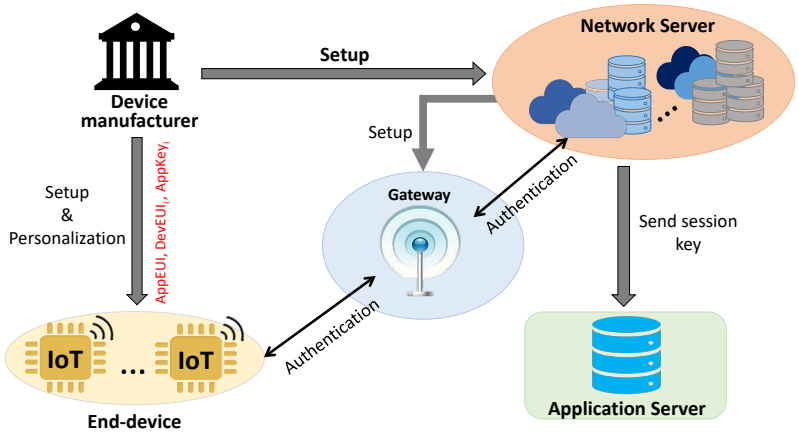


**Figure 5.** Communication between various entities in the proposed scheme.

### *3.2. Personalization*

The device manufacturer must personalize an end-device with:

- $AppEUI$: an application global unique identifier
- $DevEUI_i$: a global unique identifier of the end-device$_i$
- $AppKey_i$: a secret key shared between the end-device $_i$ and the network server

### *3.3. Authentication*

When this phase is completed successfully, end-devices connected to the same gateway share a group session key $CK$. The authentication phase consists of the following:

- Step 1: An end-device sends a join-request message to the network server to join the LoRaWAN network. The end-device$_i$
  - chooses $DevNonce_i$ at random and computes a message integrity code

$$MIC1_i = aes128\_cmac(AppKey_i, MHDR||AppEUI||DevEUI_i||DevNonce_i) \quad (6)$$

  - sends $Join\text{-}Request_i$, which contains $AppEUI, DevEUI_i, DevNonce_i$, and $MIC1_i$, to the network server.
- Step 2: To confirm the join-request, the network server sends a join-confirm message to the end-device. The network server waits until the authentication slot expires. It
  - checks the correctness of $MIC1_i$ for each join-request received during the slot.
    If $MIC1_i$ is invalid, then it *aborts*.
    Otherwise, it selects a random $r_i$ and a common random $c$.
  - It computes $a_i = E_{AppKey_i}(DevNonce_i||r_i||c)$ and sends it to end-device$_i$.
- Step 3: The end-device decrypts the join-confirm message and sends a response message to the gateway. The end-device$_i$
  - uses $AppKey_i$ to get $DevNonce_i$, $r_i$ and $c$ by decrypting $a_i$ and
  - verifies whether $DevNonce_i$ is correct.
    If $DevNonce_i$ is incorrect, it *aborts*; otherwise, it stores $c$ and sends $r_i$ to the gateway.
- Step 4: The gateway collects all the response messages at the time slot and sends a group-response message to the network server for the verification.
  - The gateway, on receiving $r_1, \cdots, r_n$ from end-devices at the time slot, sends the group-response message $g = r_1 \oplus ... \oplus r_n$ to the network server.
  - The network server computes $g' = r_1 \oplus r_2 \oplus ... \oplus r_n$ and verifies whether $g' \stackrel{?}{=} g$. If it does not hold, the gateway will send $r_1, \cdots, r_n$ to the network server and the network server will verify $r_i$ of each end-device$_i$.

- Step 5: The network server sends a join-accept message to the end-devices and sends the session keys to the application server
  - The network server computes $MIC2_i = aes128\_cmac(AppKey_i, MHDR||NetID||AppNonce_i||DevAddr_i||RFU||RxDelay_i||CFList)$ and

$$Join\text{-}Accept_i = E_{AppKey_i}(AppNonce_i||NetID||DevAddr_i||RFU||RxDelay_i||CFList||MIC2_i) \quad (7)$$

    and sends $Join\text{-}Accept_i$ to the end-device$_i$.
  - The end-device$_i$ decrypts $Join\text{-}Accept_i$ with the corresponding $AppKey_i$. Then, the end-device$_i$ verifies whether $MIC2_i$ is valid. After that, both the network server and end-device$_i$ compute a shared session key as

$$NwkSKey_i = E_{AppKey_i}(0x01||AppNonce_i||NetID||DevNonce_i||pad_{16}) \quad (8)$$
$$AppSKey_i = E_{AppKey_i}(0x02||AppNonce_i||NetID||DevNonce_i||pad_{16}) \quad (9)$$
$$CK = H(AppEUI||c) \quad (10)$$

  - Finally, the network server sends the corresponding $AppSKey_i$ and $CK$ to the application server for each end-device$_i$.

Although we use AES-128, one may use AES-256 to provide higher security. The authentication process is further discussed in Figure 6 in detail.

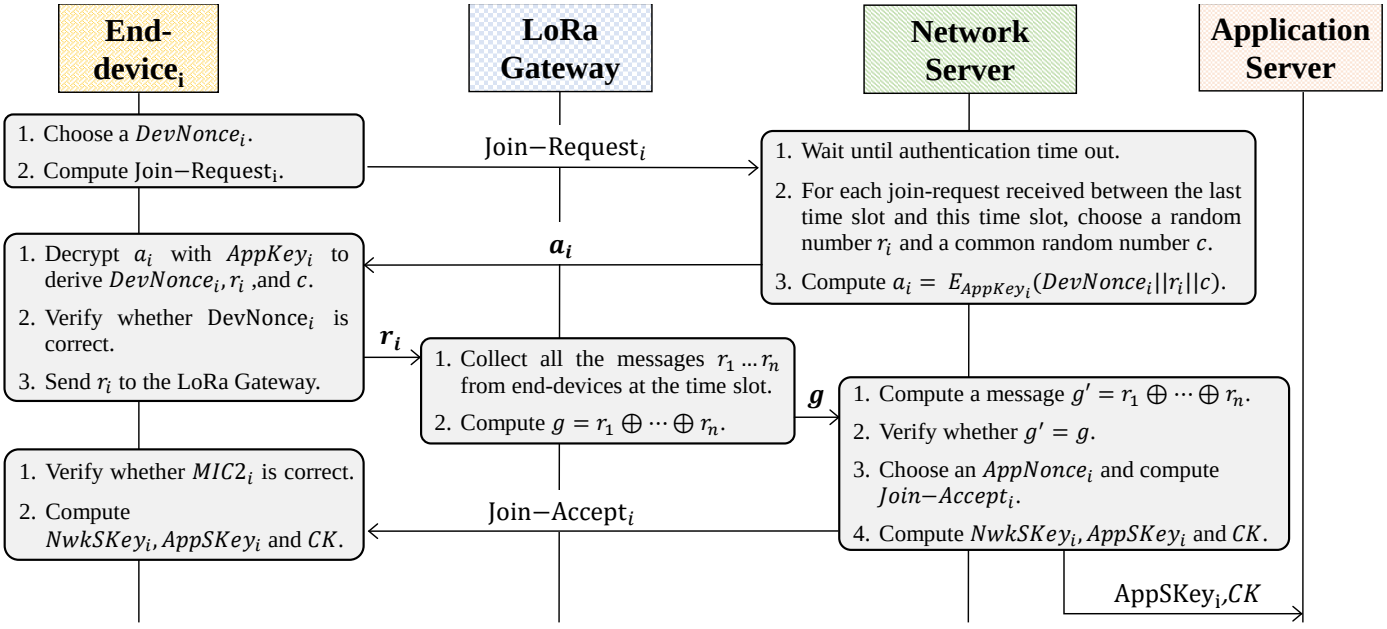


**Figure 6.** Authentication process.

## 4. Security Assurance

The proposed scheme achieves certain security features, such as multi-device authentication, resistance against session key disclosure, and strong protection against fake end-devices and fraudulent servers. Before getting into security proofs, we provide formal security models.

### *4.1. Security Model*

We provide security models for the authentication and session key agreement process. The model is a game [24] between a simulator $\mathcal{S}$ and a polynomial-time adversary $\mathcal{A}$.

1. **Secure Authentication:** $\mathcal{A}$ outputs a target $DevEUI^*$ before the game starts.
   - **Setup:** $\mathcal{S}$ sends $\mathcal{A}$ system public parameters and the symmetric functions $(E_{Key}, D_{Key})$ and sets the key and other public parameters as in the original scheme.
   - **Training:** $\mathcal{A}$ queries the two oracles as follows:
     - *Personalization:* $\mathcal{A}$ inputs $DevEUI_i$. If $DevEUI_i \neq DevEUI^*$, then $\mathcal{S}$ sends data transmitted in the personalization phase to $\mathcal{A}$. Otherwise, $\mathcal{S}$ aborts.
     - *Authentication:* $\mathcal{A}$ inputs $DevEUI_i$ and $AppEUI$. $\mathcal{S}$ sends the packages transmitted in the authentication phase to $\mathcal{A}$.
   - **Challenge:** $\mathcal{A}$ picks one of the following two cases and sends $DevEUI^*$ to $\mathcal{S}$.
     - $\mathcal{A}$ impersonates a network server in the authentication phase to pass the verification. The advantage of winning the game for $\mathcal{A}$ is $Adv^{Ser}(\mathcal{A})$.
     - $\mathcal{A}$ impersonates an end-device in the authentication phase to pass the verification. The advantage of winning the game for $\mathcal{A}$ is $Adv^{Dev}(\mathcal{A})$.
2. **Secure session key exchange:** As like earlier, $\mathcal{A}$ first outputs a target $DevEUI^*$.
   - **Setup:** $\mathcal{S}$ sends $\mathcal{A}$ system public parameters and the symmetric functions $(E_{Key}, D_{Key})$ and sets the key and other public parameters as in the original scheme.
   - **Training 1:** $\mathcal{A}$ queries the two oracles as follows.
     - *Personalization:* $\mathcal{A}$ inputs $DevEUI_i$. If $DevEUI_i \neq DevEUI^*$, then $\mathcal{S}$ sends data transmitted in the personalization phase to $\mathcal{A}$. Otherwise, $\mathcal{S}$ aborts.
     - *Authentication:* $\mathcal{A}$ inputs $DevEUI_i$ and $AppEUI$ and $\mathcal{S}$ sends packages transmitted in the authentication phase to $\mathcal{A}$.
   - **Challenge:** $\mathcal{A}$ sends $DevEUI^*$ and two random numbers $r_0$ and $r_1$. Then, $\mathcal{S}$ randomly chooses $b \in \{0, 1\}$. $\mathcal{S}$ sends $E(DevNonce^*||c||r_b)$ to $\mathcal{A}$, where $c$ is a random number chosen by $\mathcal{S}$.
   - **Training 2:** $\mathcal{A}$ queries the same queries in **Training 1**.
   - **Guess:** $\mathcal{A}$ guesses whether $b = 0$ or 1. The advantage of winning the game for $\mathcal{A}$ is defined as $Adv^{IND-CCA-SK}(\mathcal{A}) = |Pr[b = b'] - \frac{1}{2}|$.

### *4.2. Security Proof*

**Theorem 1.** *The proposed technique ensures safe mutual authentication and key exchanges between an end-device and the network server.*

**Proof.** Theorem 1 is established when Lemmas 1, 2, and 3 hold. □

**Lemma 1.** *The suggested approach is secure against an adversary impersonating the network server based on the indistinguishability of pseudorandom permutation (PRP).*

**Proof.** Assume that a polynomial-time adversary $\mathcal{A}$ impersonates the network server with a non-negligible advantage $Adv^{Ser}(\mathcal{A})$. Then, we can build a probabilistic polynomial-time simulator $\mathcal{S}$ with a non-negligible advantage $Adv^{PRP}(\mathcal{S})$ to breach the indistinguishability between a random permutation and a pseudorandom permutation. Before the setup step, $\mathcal{A}$ generates a target identity $DevEUI^*$. Let $SK$ stand for the PRP game's secret key.

- **Setup:** $\mathcal{S}$ sets the public parameters and the symmetric functions according to the pseudorandom permutation. After that, it sends the public parameters and the symmetric functions to $\mathcal{A}$.
- **Training:** $\mathcal{S}$ simulates the following oracles for $\mathcal{A}$.
  - *Personalization:* $\mathcal{A}$ sends an end-device identity $DevEUI_i$. If $DevEUI_i \neq DevEUI^*$, then $\mathcal{S}$ returns $(AppEUI, AppKey_i)$ as in the actual scheme. Otherwise, $\mathcal{S}$ aborts.

– *Authentication:* $\mathcal{A}$ sends $DevEUI_i$ and $AppEUI$. If $DevEUI_i \neq DevEUI^*$, $\mathcal{S}$ executes the authentication phase of the proposed scheme. Otherwise, $\mathcal{S}$ selects $DevNonce_i$, $c$, and $r_i$, and then queries the encryption oracle of PRP to receive ciphertext $C_1$ and $C_2$ as

$$C_1 = E_{SK}(AppEUI||DevEUI_i||DevNonce_i) \quad (11)$$
$$C_2 = E_{SK}(DevNonce_i||c||r_i) \quad (12)$$

Then, $\mathcal{S}$ selects $AppNonce_i$ and queries the encryption oracle of PRP to encrypt $NetID$, $DevAddr_i$, $RFU$, $RxDelay_i$, and $CFList$ as ciphertext $C_3$ and $C_4$, where

$$C_3 = E_{SK}(AppNonce_i||NetID||DevAddr_i||RFU||RxDelay_i||CFList) \quad (13)$$
$$C_4 = E_{SK}(AppNonce_i||NetID||DevAddr_i||RFU||RxDelay_i||CFList||C_3) \quad (14)$$

Finally, $\mathcal{A}$ receives $C_1$, $C_2$, $C_3$, and $C_4$ from the the simulator $\mathcal{S}$.

- **Challenge:** For $DevEUI_i = DevEUI^*$, $\mathcal{A}$ sends $f_1$ to $\mathcal{S}$, where

$$f_1 = E_{SK}(AppEUI||DevEUI_i||DevNonce_i)||AppEUI||DevEUI_i||DevNonce_i \quad (15)$$

Then, $\mathcal{S}$ sends $E_{SK}(AppEUI||DevEUI_i||DevNonce_i)$ to the PRP and chooses case 2 for the challenge. After that, the PRP randomly chooses $b \in \{0,1\}$. If $b = 0$, $ST$ is a random string acquired by performing the $\Omega^{-1}$ function. If $b = 1$, $ST$ is a pseudorandom string acquired by performing the decryption function $\mho^{-1}$ on $E_{SK}(AppEUI||DevEUI_i||DevNonce_i)$. As a result, the PRP game returns $ST$ to $\mathcal{S}$.
- **Guess:** On receiving $ST$, $\mathcal{S}$ checks if $ST$ is equal to $AppEUI||DevEUI_i||DevNonce_i$. If the condition is true, $\mathcal{S}$ knows that $ST$ is calculated by the decryption function $\mho^{-1}$, and thus sets $b' = 1$; otherwise, $\mathcal{S}$ sets $b' = 0$. Finally, $\mathcal{S}$ sends guess $b'$ to the PRP.

Therefore, we have $Adv^{PRP}(\mathcal{S}) = Adv^{Ser}(\mathcal{A})$, where $Adv^{PRP}(\mathcal{S})$ denotes the advantage of $\mathcal{S}$ to break the indistinguishability between a random permutation and a pseudorandom permutation. A security proof sketch is further mentioned in Figure 7. □

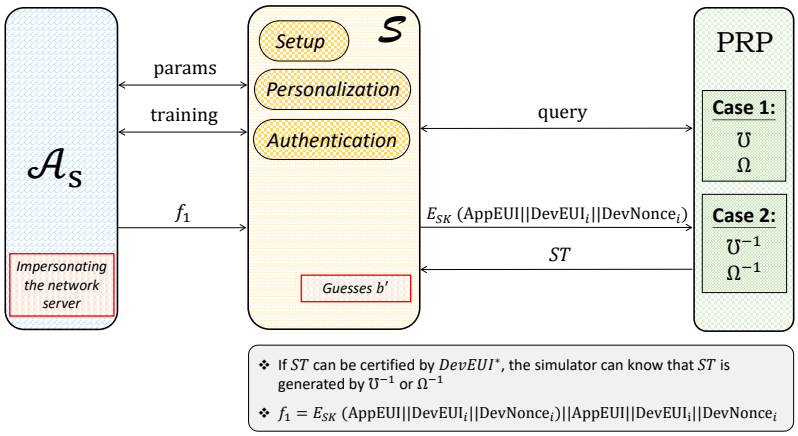


**Figure 7.** The game among $\mathcal{A}$, who impersonates the network server; $\mathcal{S}$; and PRP oracles.

**Lemma 2.** *The suggested technique is secure against an attacker impersonating a legitimate end-device based on the IND-CCA security of the symmetric encryption.*

**Proof.** Assume that there exists a polynomial-time adversary $\mathcal{A}$ who has the non-negligible advantage $Adv^{Dev}(\mathcal{A})$ to impersonate a legal end-device of the proposed scheme. Then, we can construct a probabilistic polynomial time simulator $\mathcal{S}$ that has the non-negligible advantage $Adv^{IND-CCA}(\mathcal{S})$ to break an IND-CCA symmetric encryption. The adversary

$\mathcal{A}$ outputs a target identity $DevEUI^*$ before the setup phase. Let $SK$ denote the secret key of the underlying IND-CCA symmetric encryption.

- **Setup:** $\mathcal{S}$ sets the the public parameters and the symmetric functions according to the IND-CCA symmetric encryption process. Then, $\mathcal{S}$ sends the public parameters and the symmetric functions to $\mathcal{A}$.
- **Training:** $\mathcal{S}$ simulates the following oracles for $\mathcal{A}$ as follows.
  - *Personalization:* $\mathcal{A}$ inputs an end-device identity $DevEUI_i$ to $\mathcal{S}$. If $DevEUI_i \neq DevEUI^*$, then $\mathcal{S}$ returns $AppEUI$ and $AppKey_i$ according to the proposed scheme. Otherwise, $\mathcal{S}$ aborts.
  - *Authentication:* $\mathcal{A}$ sends an end-device's identity $DevEUI_i$ and $AppEUI$. If $DevEUI_i \neq DevEUI^*$, $\mathcal{S}$ executes the authentication phase of the proposed scheme. Otherwise, $\mathcal{S}$ selects $DevNonce_i$, $c$, and $r_i$, and then queries the encryption oracle of the IND-CCA symmetric encryption to encrypt $AppEUI$, $DevEUI_i$, $DevNonce_i$, $c$ and $r_i$ into ciphertext $C_1$ and $C_2$ as

$$C_1 = E_{SK}(AppEUI||DevEUI_i||DevNonce_i) \quad (16)$$
$$C_2 = E_{SK}(DevNonce_i||c||r_i) \quad (17)$$

    Then, $\mathcal{S}$ selects $AppNonce_i$ and queries the encryption oracle of the IND-CCA symmetric encryption to encrypt system parameters: $NetID$, $DevAddr_i$, $RFU$, $RxDelay_i$, and $CFList$ to ciphertext $C_3$ and $C_4$ as

$$C_3 = E_{SK}(AppNonce_i||NetID||DevAddr_i||RFU||RxDelay_i||CFList) \quad (18)$$
$$C_4 = E_{SK}(AppNonce_i||NetID||DevAddr_i||RFU||RxDelay_i||CFList||C_3) \quad (19)$$

    Finally, $\mathcal{A}$ receives $C_1$, $C_2$, $C_3$, and $C_4$ from $\mathcal{S}$.
- **Challenge:** $\mathcal{A}$ sends identity $DevEUI_i$ to $\mathcal{S}$, where $DevEUI_i = DevEUI^*$. Then, $\mathcal{S}$ chooses $r_0$ and $r_1$ to compute $M_0 = (DevNonce_i||c||r_0)$ and $M_1 = (DevNonce_i||c||r_1)$. After that, $\mathcal{S}$ sends $(M_0, M_1)$ to the IND-CCA oracles to start the challenge phase of the IND-CCA game. Finally, $\mathcal{S}$ receives $C_b = E_{SK}(M_b)$ given from the encryption oracle and sends it to $\mathcal{A}$, where $b$, chosen by the encryption oracle, is unknown to $\mathcal{S}$.
- **Guess:** $\mathcal{A}$ outputs a bit $y_{b'}$ to $\mathcal{S}$. If $y_{b'} = 0$, $\mathcal{S}$ sets $b^{'} = 0$; otherwise, $\mathcal{S}$ sets $b^{'} = 1$. Finally, $\mathcal{S}$ sends the guess $b^{'}$ to the IND-CCA game.

Therefore, we have $Adv^{IND-CCA}(\mathcal{S}) = Adv^{Dev}(\mathcal{A})$, where $Adv^{IND-CCA}(\mathcal{S})$ denotes the advantage of the simulator to break the IND-CCA symmetric encryption. A security proof sketch is further mentioned in Figure 8. □

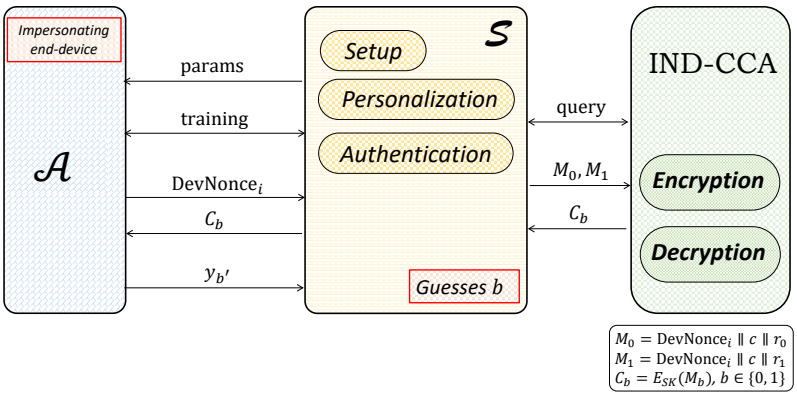


**Figure 8.** The game among $\mathcal{A}$, impersonating the network server; $\mathcal{S}$; and IND-CCA oracles.

**Lemma 3.** *The proposed method is resistant to an adversary who recovers the session key.*

**Proof.** Assume that a polynomial-time external adversary $\mathcal{A}$ has a non-negligible advantage $Adv^{\text{IND-CCA-SK}}(\mathcal{A})$ to reveal the session key. Then, we can construct a probabilistic polynomial time simulator $\mathcal{S}$ that has the non-negligible advantage $Adv^{\text{IND-CCA-SK}}(\mathcal{S})$ to break a known symmetric encryption with IND-CCA security.

- **Setup:** $\mathcal{S}$ sets the the public parameters and the symmetric functions according to the IND-CCA symmetric encryption. Then, $\mathcal{S}$ sends the public parameters and the symmetric functions to $\mathcal{A}$.
- **Training 1:** $\mathcal{S}$ simulates the following oracles for $\mathcal{A}$ as follows.
  - *Personalization:* $\mathcal{A}$ sends an end-device identity $DevEUI_i$. If $DevEUI_i \neq DevEUI^*$, then $\mathcal{S}$ returns $AppEUI$ and $AppKey_i$, according to the proposed scheme. Otherwise, $\mathcal{S}$ aborts.
  - *Authentication:* $\mathcal{A}$ sends identity $DevEUI_i$ and $AppEUI$ to S. If $DevEUI_i \neq DevEUI^*$, $\mathcal{S}$ executes the authentication phase of the proposed scheme. Otherwise, $\mathcal{S}$ selects $DevNonce_i$, $c$ and $r_i$ and then queries the encryption oracle of the IND-CCA symmetric encryption to encrypt $AppEUI$, $DevEUI_i$ and $DevNonce_i$ to ciphertext $C_1 = E_{SK}(AppEUI||DevEUI_i||DevNonce_i)$ and encrypts $c$ and $r_i$ to ciphertext $C_2 = E_{SK}(DevNonce_i||c||r_i)$. Once, $\mathcal{S}$ receives $(C_1, C_2)$, it selects $AppNonce_i$ and queries encryption oracle of the IND-CCA symmetric encryption for $(NetID, DevAddr_i, RFU, RxDelay_i, CFList)$, and gets ciphertext $(C_3, C_4)$ as

$$C_3 = E_{SK}(AppNonce_i||NetID||DevAddr_i||RFU||RxDelay_i||CFList) \quad (20)$$

$$C_4 = E_{SK}(AppNonce_i||NetID||DevAddr_i||RFU||RxDelay_i||CFList||C_3) \quad (21)$$

    Finally, $\mathcal{A}$ receives $C_1$, $C_2$, $C_3$ ,and $C_4$ from $\mathcal{S}$.
- **Challenge:** $\mathcal{A}$ sends $r_0$ and $r_1$ of equal-length and $DevEUI_i$, where $DevEUI_i = DevEUI^*$. Then, $\mathcal{S}$ computes $M_0 = (DevNonce_i||c||r_0)$ and $M_1 = (DevNonce_i||c||r_1)$ and sends $(M_0, M_1)$ to the IND-CCA oracles to start the challenge phase of the underlying IND-CCA game. After that, $\mathcal{S}$ receives $C_b = E_{SK}(M_b)$ given from the encryption oracle and sends it to $\mathcal{A}$, where $b \in_R \{0, 1\}$ is unknown to $\mathcal{S}$.
- **Training 2:** $\mathcal{A}$ can query the same queries as those in **Training 1**.
- **Guess:** The adversary $\mathcal{A}$ outputs a bit $y_{b'}$ to $\mathcal{S}$. If $y_{b'} = 0$, $\mathcal{S}$ sets $b' = 0$; otherwise, $\mathcal{S}$ sets $b' = 1$. Finally, $\mathcal{S}$ sends the guess $b'$ to the IND-CCA game.

Therefore, we have $Adv^{\text{IND-CCA-SK}}(\mathcal{S}) = Adv^{\text{IND-CCA-SK}}(\mathcal{A})$, where $Adv^{\text{IND-CCA-SK}}(\mathcal{S})$ denotes the advantage of the simulator to break the IND-CCA symmetric encryption. A security proof sketch is further mentioned in Figure 9. □

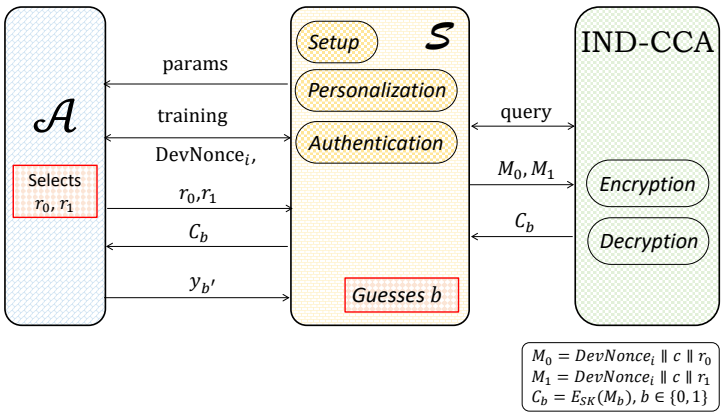


**Figure 9.** The security game among $\mathcal{A}$, who can obtain the session key; $\mathcal{S}$; and IND-CCA oracles.

## 5. Comparison

This section demonstrates the proposed scheme's characteristics, computation cost, and performance.

### *5.1. Security Properties*

The proposed scheme's characteristics are compared to the LoRaWAN specifications in Table 2. The suggested scheme and the LoRaWAN specifications ensure mutual authentication and data integrity. LoRaWAN only checks one end-device per time slot, whereas the suggested scheme certifies several end-devices at the same time.

**Table 2.** Functionality comparison.

| Security Properties | LoRaWAN Spec. | The Proposed Scheme |
|---|---|---|
| Mutual Authentication | Yes | Yes |
| Data Integrity Protection | Yes | Yes |
| Authentication of Multiple End-Devices | No | Yes |

### *5.2. Computation Cost*

The computation overhead considers the additional cost incurred due to rapid authentication for multiple end-devices in LoRaWAN. We list the hardware information and the computation cost of each operation in Table 3.

**Table 3.** Device specification and operation overhead.

| (**a**) The hardware information. | |
|---|---|
| **Specification** | **Windows 10 64-bit** |
| CPU | Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz |
| RAM | 8.00 GB |
| Motherboard | KBL Strongbow KL |
| (**b**) Benchmark time. | |
| **Operation** | **Time(s)** |
| AES [25] encryption /decryption | 0.0209808 |
| XOR operation | 0.0009536 |

In Table 4, the computation cost of the LoRaWAN specifications is compared to the overhead of the proposed method. Although the suggested system involves more computations in the gateway and network server than the LoRaWAN specifications, it has the batch authentication property that the LoRaWAN specifications do not have.

**Table 4.** Computation cost.

| Entity | LoRaWAN Specification | The Proposed Scheme |
|---|---|---|
| Each End-device | $3T_{AES} \approx 0.06294\ s$ | $4T_{AES} \approx 0.08392\ s$ |
| Gateway | — | $nT_{xor} \approx 0.00095n\ s$ |
| Network Server | $3nT_{AES} \approx 0.06294n\ s$ | $4nT_{AES} + nT_{xor} \approx 0.08487n\ s$ |

It is worth noting that the authentication computation cost does not include the cost of computing the shared session key.

### *5.3. Performance*

Based on the work of Lavric and Popa [26] consideration, we set the maximum number of end-devices in the LoRaWAN network to be 1000. Furthermore, we chose these scenarios for comparison since 50 or 100 end-devices are commonly utilized in real-life. Table 5 shows the overall execution time of the proposed scheme and of the LoRaWAN specifications for validating 50, 100, and 1000 end-devices, respectively.

**Table 5.** Performance comparison.

| Number of Devices | Overhead | | Efficacy |
|---|---|---|---|
| | LoRaWAN Spec. | The Proposed Scheme | |
| 50 | $\approx$ 06.294 *s* | $\approx$ 04.327 *s* | 31% |
| 100 | $\approx$ 12.588 *s* | $\approx$ 08.571 *s* | 32% |
| 1000 | $\approx$ 125.880 *s* | $\approx$ 84.954 *s* | 33% |

Because the LoRaWAN specifications check one device attempting to join the network at a time, the joining method must be repeated 50, 100, or 1000 times in order to verify all end-devices, correspondingly, as shown in Table 5. On the other hand, since the suggested scheme provides batch verification for multiple devices, it only has to be run once in each case to verify all end-devices.

## 6. Conclusions

Based on the LoRaWAN join procedure, we have presented a secure multi-device authentication scheme. The suggested system can fit the features of natural LoRaWAN environments to reduce join latency. Furthermore, the suggested technique employs the challenge-response and exclusive-OR operations in tandem to obtain batch authentication benefits. As a result, the suggested scheme may be readily integrated into the existing LoRaWAN join mechanism. Furthermore, we anticipate that the suggested approach can be utilized to create a secure and rapid LoRaWAN authentication system that meets the LoRa Alliance's specifications. In the future, we will consider various real-world scenarios and design effective key management and device revocation processes in multi-device authentication settings.

**Author Contributions:** Conceptualization, C.-I.F. and C.-H.S.; methodology, C.-I.F., E.-S.Z., A.K., C.-H.S.; software, C.-H.S.; validation, C.-I.F., E.-S.Z. and A.K.; formal analysis, E.-S.Z. and A.K.; investigation, E.-S.Z., A.K. and C.-H.S.; resources, C.-I.F. and C.-H.S.; data curation, C.-H.S.; writing—original draft preparation, E.-S.Z. and C.-H.S.; writing—review and editing, C.-I.F., E.-S.Z., and A.K.; visualization, A.K. and C.-H.S.; Supervision, C.-I.F. and A.K.; project administration, C.-I.F.; funding acquisition, C.-I.F. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by the Ministry of Science and Technology of Taiwan under grants 110-2218-E-110-007-MBK and MOST 109-2221-E-110-044-MY2.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Acknowledgments:** This research was partially supported by Taiwan Information Security Center at National Sun Yat-sen University (TWISC@NSYSU). It was also supported by the Information Security Research Center at National Sun Yat-sen University in Taiwan and the Intelligent Electronic Commerce Research Center from The Featured Areas Research Center Program within the Higher Education Sprout Project framework by the Ministry of Education (MOE) in Taiwan.

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

## References


1. Basford, P.J.; Bulot, F.M.; Apetroaie-Cristea, M.; Cox, S.J.; Ossont, S.J. LoRaWAN for smart city IoT deployments: A long term evaluation. *Sensors* **2020**, *20*, 648. [CrossRef] [PubMed]
2. Lima, E.; Moraes, J.; Oliveira, H.; Cerqueira, E.; Zeadally, S.; Rosário, D. Adaptive priority-aware LoRaWAN resource allocation for Internet of Things applications. *Ad Hoc Netw.* **2021**, *122*, 102598. [CrossRef]
3. Ericsson, S. Internet of Things to Overtake Mobile Phones by 2018. Available online: http://www.satellitemarkets.com/market-trends/internet-things-overtake-mobile-phones-2018 (accessed on 2 March 2022).
4. Statista, I. Internet of Things (IoT) Connected Devices Installed Base Worldwide from 2015 to 2025 (In Billions). Available online: https://statinvestor.com/data/33967/iot-number-of-connected-devices-worldwide/ (accessed on 2 March 2022).
5. Alliance, L. *LPWA Technologies Unlock New IoT Market Potential*; White Paper; LoRa Alliance: Fremont, CA, USA, 2015. Available online: https://lora-alliance.org/resource_hub/lorawan-security-whitepaper/ (accessed on 2 March 2022) .
6. Sinha, R.S.; Wei, Y.; Hwang, S.H. A survey on LPWA technology: LoRa and NB-IoT. *ICT Express* **2017**, *3*, 14–21.
7. Navarro-Ortiz, J.; Sendra, S.; Ameigeiras, P.; Lopez-Soler, J.M. Integration of LoRaWAN and 4G/5G for the Industrial Internet of Things. *IEEE Commun. Mag.* **2018**, *56*, 60–67. [CrossRef]
8. Noura, H.; Hatoum, T.; Salman, O.; Yaacoub, J.P.; Chehab, A. LoRaWAN security survey: Issues, threats and possible mitigation techniques. *Internet Things* **2020**, *12*, 100303. [CrossRef]
9. Adelantado, F.; Vilajosana, X.; Tuset-Peiro, P.; Martinez, B.; Melia-Segui, J.; Watteyne, T. Understanding the limits of LoRaWAN. *IEEE Commun. Mag.* **2017**, *55*, 34–40. [CrossRef]
10. Semtech. Why Lora®? Available online: https://www.semtech.com/lora/why-lora (accessed on 2 March 2022).
11. Bocker, S.; Arendt, C.; Jorke, P.; Wietfeld, C. LPWAN in the Context of 5G: Capability of LoRaWAN to Contribute to mMTC. In Proceedings of the 2019 IEEE 5th World Forum on Internet of Things (WF-IoT), Limerick, Ireland, 15–18 April 2019; pp. 737–742.
12. Cisco. Cisco Solution for LoRaWAN. Available online: https://www.cisco.com/c/en/us/solutions/internet-of-things/lorawan-solution.html (accessed on 2 March 2022).
13. LoRa Alliance. *A Technical Overview of LoRa and LoRaWAN*; LoRa Alliance: Fremont, CA, USA, 2015. Available online: https://www.everythingrf.com/whitepapers/details/2682-a-technical-overview-of-lora-and-lorawan (accessed on 2 March 2022).
14. Butun, I.; Pereira, N.; Gidlund, M. Security risk analysis of LoRaWAN and future directions. *Future Internet* **2019**, *11*, 3. [CrossRef]
15. Lombardo, A.; Parrino, S.; Peruzzi, G.; Pozzebon, A. LoRaWAN vs NB-IoT: Transmission Performance Analysis within Critical Environments. *IEEE Internet Things J.* **2021**, *9*, 1068–1081. [CrossRef]
16. Chen, X.; Wang, J.; Wang, L. A fast session key generation scheme for LoRaWAN. In Proceedings of the 2019 Australian & New Zealand Control Conference (ANZCC), Auckland, New Zealand, 27–29 November 2019; pp. 63–66.
17. Danish, S.M.; Lestas, M.; Asif, W.; Qureshi, H.K.; Rajarajan, M. A lightweight blockchain based two factor authentication mechanism for LoRaWAN join procedure. In Proceedings of the 2019 IEEE International Conference on Communications Workshops (ICC Workshops), Shanghai, China, 20–24 May 2019; pp. 1–6.
18. Tsai, K.L.; Leu, F.Y.; Hung, L.L.; Ko, C.Y. Secure session key generation method for LoRaWAN servers. *IEEE Access* **2020**, *8*, 54631–54640. [CrossRef]
19. Jabbari, A.; Mohasefi, J.B. A Secure and LoRaWAN Compatible User Authentication Protocol for Critical Applications in the IoT Environment. *IEEE Trans. Ind. Inform.* **2021**, *18*, 56–65. [CrossRef]
20. Kaven, S.; Bornholdt, L.; Skwarek, V. Authentication by rssi-position based localization in a lora lpwan. In Proceedings of the 2020 6th IEEE Congress on Information Science and Technology (CiSt), Agadir, Morocco, 5–12 June 2021; pp. 448–454.
21. Gu, C.; Jiang, L.; Tan, R.; Li, M.; Huang, J. Attack-aware synchronization-free data timestamping in lorawan. *ACM Trans. Sens. Netw. (TOSN)* **2021**, *18*, 1–31. [CrossRef]
22. Ertürk, M.A.; Aydın, M.A.; Büyükakkaşlar, M.T.; Evirgen, H. A survey on LoRaWAN architecture, protocol and technologies. *Future Internet* **2019**, *11*, 216. [CrossRef]
23. Sornin, N.; Luis, M.; Eirich, T.; Kramp, T.; Hersent, O. *Lorawan Specification*; LoRa Alliance: Fremont, CA, USA, 2015. Available online: https://lora-alliance.org/resource_hub/lorawan-specification-v1-1/ (accessed on 2 March 2022)
24. Eldefrawy, M.; Butun, I.; Pereira, N.; Gidlund, M. Formal security analysis of LoRaWAN. *Comput. Netw.* **2019**, *148*, 328–339. [CrossRef]
25. Draft, F. *Advanced Encryption Standard (AES)*; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2001 .
26. Lavric, A.; Popa, V. Performance evaluation of LoRaWAN communication scalability in large-scale wireless sensor networks. *Wirel. Commun. Mob. Comput.* **2018**, *2018*, 1–10. [CrossRef]