# Security Enhancement of OTAA based Joining Procedure in LoRaWAN for Satellite Communication

Jay Dave*, Nikumani Choudhury*
* Dept. of Computer Science & Information Systems, BITS, Pilani, Hyderabad Campus, India
Email: {jay.dave, nikumani}@hyderabad.bits-pilani.ac.in

***Abstract*—Long Range Wide Area Network (LoRaWAN) is a wireless communication protocol that facilitates efficient and wide-range communication under low-power conditions. Over-the-air activation (OTAA) is a process recommended by LoRaWAN v1.0.4 (latest version) that enables end devices to join the network with the help of the joining server and generate the session keys for further communications. However, OTAA is vulnerable to potential security threats due to unencrypted join request messages and the reuse of the same encryption keys. In this paper, we present a new security enhancement that addresses the aforesaid security issues. In the proposed scheme, we encrypt the join request message using a secret key to ensure data confidentiality. In addition, we incorporate the use of a random nonce in the proposed joining procedure to protect LoRaWAN against attacks related to the reuse of the same key. We show that the adversary cannot learn sensitive information from the join request message and reuse the $AppKey$ to execute the eavesdropping and unauthorized activation attacks with non-negligible probability.**

***Index Terms*—Long Range Wide Area Network (LoRaWAN), Encryption, Over-The-Air-Activation (OTAA), Satellite Communication.**

## I. Introduction

Internet of Things (IoTs) are becoming pervasive and a fundamental business interest in global commerce. LoRaWAN, a low-power and wide-area network protocol, offers a scalable and resilient solution for interconnecting IoT devices over a large geographical coverage. This technology caters to the distinct requirements of IoT applications by enabling wireless data transmission from sensors and devices to the network. The long-range capacities, low power usage, and support for various IoT use cases of LoRaWAN render it an optimal selection for interconnecting actuators, sensors, and other IoT devices [1]. The LoRaWAN architecture includes of five main entities: end devices, gateways, network server, join server, and application server.

Over-the-air activation (OTAA) is a vital notion of LoRaWAN, playing an important role in securely joining end devices to a network [1]. IoT devices start the process by sending a joining request to the join server through the gateway server. The join server responds to the end device if it is eligible to join the network. The end device learns the factors from the join response. These factors are used to generate the session keys, which are applicable in forthcoming communications.

Although LoRaWAN v1.0.4 considers OTAA as a secure algorithm, OTAA is vulnerable to serious security issues. Firstly, according to OTAA, the end device will send the join request containing the sensitive information in unencrypted form. As a result, an adversary may intercept the communication and learn the sensitive information. The attacker may exploit the information to execute a replay attack or man-in-the-middle attack. Secondly, the app key (i.e., a cryptographic encryption key) is used by the end device and the join server to derive the session keys. Nevertheless, reusing a single key across multiple sessions could give rise to significant security vulnerabilities. Specifically, the adversary who successfully obtains the $AppKey$, can exploit the key to expose sensitive information through eavesdropping or perform unauthorized activation.

In this paper, we propose a new security advancement for OTAA to resolve the aforesaid security concerns. In our approach, we encrypt the join request using $AppKey$. As the key is secretly shared between the end device and the join server, it limits the access of the join request's content to only the legitimate end device and the join server. Additionally, we include a random nonce along with $DevNonce$ in the proposed approach. Hence, the join server, which knows $DevNonce$ and $AppKey$, is able to retrieve the random nonce from the join request message. The random nonce, along with $AppKey$, is used to derive the session keys.

## II. Background and Problem Statement

In this section, we briefly discuss LoRaWAN architecture and OTAA. Thereafter, we explain the security issues in OTAA.

### A. LoRaWAN Architecture

A low-power, wide-area networking protocol, LoRaWAN is constructed using the LoRa radio modulation method. It connects devices to the network wirelessly and facilitates communication between network gateways and end-node devices. LoRaWAN v1.0.4 refers to the latest version that was released in October 2020. It potentially introduces new features, bug corrections, and other enhancements as compared to its predecessors. LoRaWAN architecture is intended to enable low-power IoT devices to communicate efficiently over extended distances. The primary components of the proposed

979-8-3503-8741-4/24/$31.00 ©2024 IEEE

architecture are as follows: End Devices ($ED$), Satellite Gateways ($SG$), Network Server ($NS$), Join Server ($JS$), and Application Server ($AS$).

- *End Devices ($ED$)*: They are the devices that send data to the gateway through LoRa network. $ED$ comprises the following components: (1) sensors, which measure environmental parameters including pressure, temperature, vibration, humidity, velocity, etc.; (2) a LoRa transceiver, which enables wireless communication via LoRa modulation mechanism; and (3) microcontrollers, which oversee data processing, communication management, and device functions.
- *Satellite Gateways ($SG$)*: They act as intermediaries between the end devices and the network server. SG establishes communication with $ED$ via LoRaWAN technologies and with the network server via backhaul technologies, including ethernet, Wi-Fi, and cellular technology.
- *Network Server ($NS$)*: It coordinates the inter-device (IoT device) communication with the other parts of the network. The functionalities of $NS$ include validating and processing uplink messages (i.e., data sent by $ED$s through $SG$), managing device activation with the help of the join server, and routing downlink messages (i.e., application server's commands/responses to end devices).
- *Join Server ($JS$)*: It coordinates with $NS$ to ensure the secure joining of new end devices. Additionally, $JS$ generates and distributes session keys for newly joined $ED$s.
- *Application Server ($AS$)*: It is responsible for managing and processing data obtained from $ED$s.

### B. Over-the-Air Activation (OTAA)

It is an essential procedure in LoRaWAN that facilitates the secure connection of end devices to the LoRaWAN network. $ED$ initiates the process by submitting an "Join-Request" message. After validating the request, $NS$ responds with a "Join-Accept" message. $ED$ and $JS$ use the parameters of "Join Accept" to derive the session keys. These key are used to encrypt the payload during further communication with $NS$ and $AS$. In the following, we discuss the steps of OTAA process in detail.

1) $ED$ sends a "Join-Request" to $NS$. The request message includes $JoinEUI$, $DevEUI$, and $DevNonce$.
   $JoinEUI$ is an 8-byte universally unique identifier assigned to the join server by the network operator from IEEE EUI64 address space. $DevEUI$ is an 8-byte universally unique identifier assigned to the end device by the device manufacturer from IEEE EUI64 address space. $DevNonce$ is a 2-byte counter started from 0 and increments with every Join-Request. The value of $DevNonce$ is stored by $ED$ and $NS$ in their non-volatile memory. $DevNonce$ cannot be reused or changed. $JS$ rejects "Join-Request", if $DevNonce$ is not correct.
   As per the latest version "LoRaWAN v1.0.4", Join-Request is not encrypted.
2) $JS$ validates the received request. If $ED$ is eligible to join the network, $JS$ responds with "Join-Accept". The accept message contains: $JoinNonce$, $NetID$, $DevAddr$, $DLSettings$, $RXDelay$, and $CFList$.
   $JoinNonce$ is a 3-byte distinct value provided by $JS$. $NetID$ is a 3-byte network identifier value. $DevAddr$ is a 4-byte distinct value allotted by the server to each end device that is connected to it. $DLSettings$ and $RXDelay$ are downlink configuration settings and delay-related values, respectively. $CFList$ stands for channel frequency list values.
   $JS$ derives the session keys as follows.

$$NwkSKey = AES128\_encrypt(AppKey, 0 \times 01|JoinNonce|NetID|DevNonce|pad_{16}) \quad (1)$$

$$AppSKey = AES128\_encrypt(AppKey, 0 \times 02|JoinNonce|NetID|DevNonce|pad_{16}) \quad (2)$$

3) $ED$ generates $NwkSKey$ and $AppSKey$ using $JoinNonce$ and $NetID$ obtained from Join-Accept as discussed in equations 1 and 2.

### C. Security Issues in OTAA

OTAA algorithm faces the following potential security issues related to unencrypted Join Requests.

- *Exposure of DevEUI*: An attacker may easily reveal $ED$'s unique identifier by intercepting the communication between the $ED$ and $JS$ because the request message is unencrypted. S/he may exploit this information to execute the impersonation attack.
- *Replay attack*: For disrupting the upcoming joining processes, an adversary may eavesdrop on the join request and replay it by modifying $DevNonce$ value. Whenever $ED$ sends the join request again for the subsequent session, $JS$ discards it as the received $DevNonce$ value does not match the stored value.
- *Man-in-the-Middle Attacks*: An adversary may intercept the unencrypted request, alter its contents, and forward the modified request to $JS$. It may result in the activation of an unauthorized device.

The reuse of $AppKey$ by an end device presents the following security vulnerabilities:

- *Activation of unauthorized device*: An attacker who obtains $AppKey$ may manipulate it to activate illegitimate instances of the compromised device on the network.
- *Eavesdropping*: An adversary may eavesdrop and decrypt communications associated with the compromised device when $AppKey$ is compromised.

## III. Proposed Approach

In this section, we propose a new security enhancement in OTAA. The proposed scheme addresses the security issues of OTAA as mentioned in Section II-C. The proposed approach is depicted in Figure III.

1) LoRaWAN v1.0.4 assumes that the end device ($ED$) has prior knowledge about $DevEUI$, $JoinEUI$, $AppKey$, and $DevNonce$. As described in Section II-B, $DevEUI$ is a unique identifier of each end device, and it is assigned by the device manufacturer. The unique identifier of the join server ($JS$) is denoted by $JoinEUI$. $AppKey$ is a cryptographic key that is shared in secret between $JS$ and $ED$.
   Initially, $ED$ generates a pseudorandom value $R_{seed}$ using a secure and lightweight pseudorandom generator. It computes $DevNonce \oplus R_{seed}$. $ED$ generates the Join-Request frame that includes $JoinEUI$, $DevEUI$, and $DevNonce \oplus R_{seed}$. To ensure confidentiality, $ED$ encrypts the Join-Request using $AppKey$ and AES encryption algorithm in ECB mode. $ED$ sends the encrypted join request to the join server via the gateway server.
2) $JS$ decrypts the received request using $AppKey$ and AES decryption algorithm in ECB mode. Then, the server checks whether the requesting end-device is permissible to join the network. If $ED$ is eligible, $JS$ extracts the random value $R_{seed}$ from $DevNonce \oplus R_{seed}$ as the server already knows $DevNonce$ value corresponding to this device. $JS$ uses $R_{seed}$ and secure PRNG and derives a random values $R_{val}$. The server derives the session keys as follows.

$$\begin{aligned} &NwkSKey \\ &= AES128\_encrypt(AppKey \oplus R_{val}, \\ &0 \times 01 | JoinNonce | NetID | DevNonce | pad_{16}) \end{aligned} \tag{3}$$

$$\begin{aligned} &AppSKey \\ &= AES128\_encrypt(AppKey \oplus R_{val}, \\ &0 \times 02 | JoinNonce | NetID | DevNonce | pad_{16}) \end{aligned} \tag{4}$$

   $JS$ geneartes Join-Accept frame that consists of $JoinNonce$, $NetID$, $DevAddr$, $DLSettings$, $RXDelay$, and $CFList$. The server encrypts the accept message using AES algorithm in ECB mode as described in LoRaWAN v1.0.4. $JS$ sends the encrypted accept message to $ED$.
3) $ED$ decrypts the received join accept message using AES algorithm in ECB mode. The device computes the session keys using equation 3 and 4.

## IV. Security Analysis

In this section, we analyze the security and performance of the proposed approach.

*Security issues related to unencrypted Join Requests*

As discussed in Section II-C, the adversary may obtain $DevEUI$, $JoinEUI$, and $DevNonce$ by intercepting the unencrypted join request in OTAA. Using this information, the adversary may implement the impersonation attack. Moreover, the attacker may disrupt the system by replaying the intercepted message with the altered $DevNonce$. Consequently, the join server ($JS$) declines the subsequent request sent by the genuine end device ($ED$) due to a discrepancy between the received value of $DevNonce$ and the stored value. The adversary may exploit the unencrypted join request to conduct the man-in-the-middle attack.

For protection against the aforementioned security problems, we proposed the encryption for the join request. In particular, $ED$ encrypts the join request using $AppKey$ and AES encryption algorithm in ECB mode. Therefore, only $JS$ can decrypt the join request message using AES decryption algorithm because only $JS$ and genuine $ED$ know $AppKey$. The probability of an adversary to correctly guess $AppKey$ is:

$$Pr(Succ - 1) = 2^{-128} + \epsilon_1(n) \tag{5}$$

Since $AppKey$ is a randomly generated key, $\epsilon_1(n)$ is negligible. In this way, the attacker cannot decrypt the join request and learn the sensitive information with non-negligible probability. If s/he captures and replays the same request message, $JS$ rejects the request due to the repetition of the same $DevNonce$ value.

*Security issues related to reuse of AppKey*

$AppKey$ is a root key used to encrypt the communications between $ED$ and $JS$ and generate the session keys. As discussed in Section II-C, if an attacker successfully obtains $AppKey$, s/he can easily eavesdrop the communication and learn the sensitive content. Furthermore, the attacker may exploit the $AppKey$ for activating the illegal instances of the victim device on the network.

For protection against the aforementioned attacks, we deploy a random nonce along with $AppKey$ as discussed in Section III. $AppKey \oplus R$ is used to encrypt/decrypt the join accept message and generate the session keys. The probability for an adversary to correctly guess the value of $R$ is:

$$Pr(Succ_2) = 2^{-128} + \epsilon_2(n) \tag{6}$$

Since $R$ is generated using a secure PRNG, $\epsilon_2(n)$ is negligible. In this way, the adversary cannot reuse the $AppKey$ to execute the attacks with non-negligible probability.

## V. Related Work

A study conducted by Oniga et al. [2] examined the security dimensions of LoRaWAN. The authors implemented the scenarios of the security attacks such as Sniffing, Impersonation, Replay, False modification in communication parameters, Man-in-the-middle attack, DOS, Intrusion, and malware. The authors also discussed the recommendations to protect LoRaWAN against the discussed security attacks. In [3], Butun et al. provided a comprehensive study about

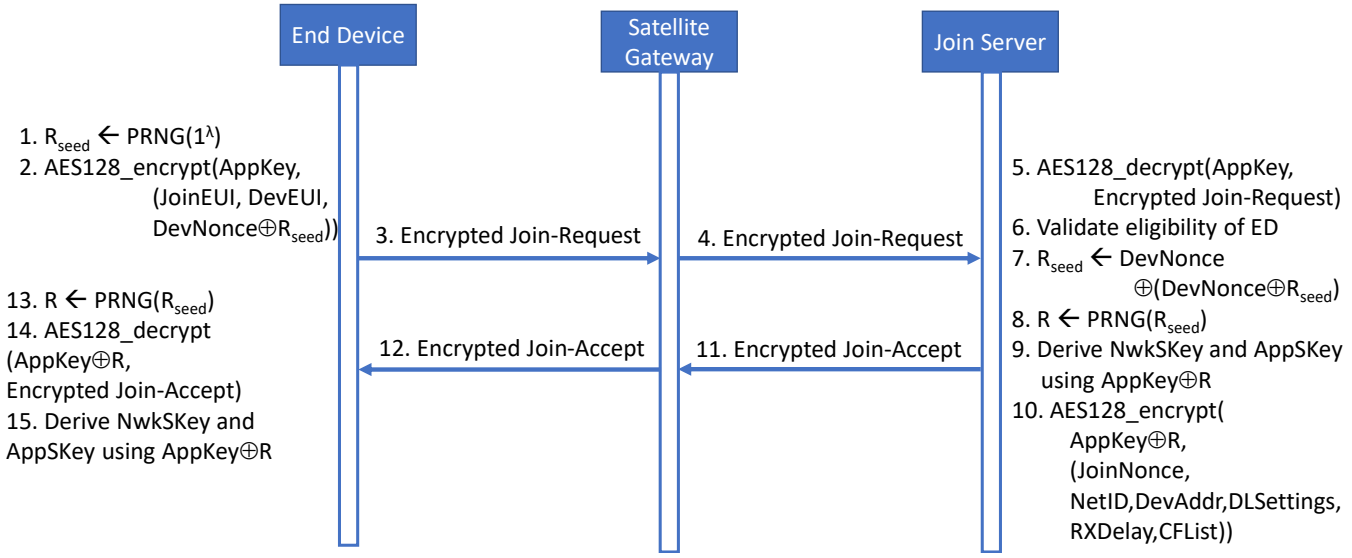


Fig. 1. Proposed Approach

LoRaWAN v1.1 and its security vulnerabilities. For OTAA in LoRaWAN v1.1, the authors discussed security threats, including replay attacks, man-in-the-middle attacks, traffic analysis, beacon synchronization, and RF interference. Furthermore, the authors deliberated on potential countermeasures against the aforementioned threats, including the use of dynamic session states, the implementation of secure key management, and the consideration of backward compatibility implications.

Eldefrawy et al. [4] presented formal security studies of LoRaWAN versions 1.0 and 1.1. The authors analyzed the security flaws in the key exchange procedure utilizing the Scyther utility. A vulnerability in LoRaWAN v1.0 was identified in its lack of synchronization, which rendered it susceptible to replay attacks. Nevertheless, the authors were unable to identify any security vulnerabilities in LoRaWAN v1.1 due to the limitations of the Scyther tool. The study does not consider real-world implementation and deployment scenarios, which may introduce additional security vulnerabilities. Moreover, the research does not consider important security aspects such as data encryption and integrity. Fehri et al. [5] proposed an experimental investigation of OTAA that examines collisions, retransmissions, and the joining process cycle. Kuntke et al. [6] presented a review of LoRaWAN related security issues in the context of IoT systems in agriculture. The security vulnerabilities associated with LoRaWAN were examined by Naidu et al. [7], including those pertaining to physical devices, DoS attacks, bit reversal attacks, ACK spoofing, energy exhaustion, and key-related attacks.

Yang et al. [8] implemented the five attacks against LoRaWAN v1.0.2: Battery exhaustion attack, Replay attack, Eavesdropping, Message falsification, and Alternation in acknowledgment packets. The authors proposed the security measures to defend against these attacks: (1) Counter value management and rekeying to safeguard against eavesdropping, (2) Secure joining process, physical protection, and rekeying to prevent replay attacks, (3) Authenticated encryption and hash value inclusion to prevent unauthorized modification of message and acknowledgment packets. Butun et al. [9] conducted an extensive investigation into the security vulnerabilities of LoRaWAN v1.1, including bit flipping, fraudulent packets, flooding, jamming attacks, and man-in-the-middle attacks. The attacks are classified as follows by the authors: (1) MITM attack, (2) Physical attack, (3) Dishonest gateway attack, and (4) Routing attack. Secure key management, secure nonce, and counter management, secure authentication schemes, and the implementation of tamper-resistant hardware were the security measures proposed by the authors.

Mårlind et al. [10] proposed a public key cryptography-based OTAA scheme. The suggested methodology enables the devices to derive the updated key periodically through the utilization of an elliptic curve algorithm. Ribeiro et al. [11], [12] proposed a blockchain-based key management scheme for LoRaWAN. The authors assumed that the joining server is a single point of failure because it is responsible for storing and managing all keys. As a solution, Ribeiro et al. deployed a blockchain architecture in conjunction with the joining server to manage the key securely. In practice, however, the centralized nature of the joining server may be incompatible with the decentralized architecture of the blockchain. Noura et al. [13] discussed a detailed survey of security issues and mitigation techniques for LoRaWAN. The authors elaborated on LoRaWAN architecture and in-built security features. Moreover, security vulnerabilities such as authentication attacks, availability attacks, confidentiality attacks, and integrity attacks were discussed [13]. Hess et al. [14] propose a firmware update server that operates in conjunction with ChirpStack. Noura et al. [15] proposed counter-based and physical channel-based solutions against

eavesdropping and replay attacks on ABP protocol.

Milani et al. [16] introduced a rejoining procedure based on a public key for OTAA. Given that OTAA prohibits the modification of root keys, the authors employed elliptic curve methodology to construct an additional layer. Barriga et al. [17] analyzed the authentication issues on the gateway using the Scyther tool. The authors proposed a session key management protocol to protect the authentication scheme. The jamming attack on OTAA was discussed in [18]. Fujdiak et al. [19] compare the performance, security, and cost of private and public LoRaWAN deployments. The comparison is conducted in three dimensions: communication performance, security, and cost analysis. Sujatha et al. [20] presented a modified elliptic curve based and artificial flora based key generation scheme for the joining process in OTAA. This modified algorithm improves performance in terms of latency, throughput, and timeout messages. Abboud et al. [21] demonstrates that augmenting the key size from 128 to 256 bits significantly enhances the resilience of LoRaWAN against various cyber attacks. Dave et al. presented new ownership verification schemes for cloud based architecture in [22], [23]. The secure encryption schemes for deduplication architecture were introduced in [24], [25]. In [26], Dave et al. proposed a new key management scheme for IoT systems.

## VI. Conclusion

In this paper, we address the potential security risks of OTAA due to unencrypted join requests and reuse of the same $AppKey$. In particular, the adversary can eavesdrop on the join request and exploit the sensitive information to execute replay and man-in-the-middle attacks in OTAA. We propose encrypting the join request to protect against these security risks. It restricts the access of request messages to legitimate entities. Next, the end device reuses the same $AppKey$ in OTAA. Hence, the attacker who captures the $AppKey$ is able to eavesdrop on the communication and execute the false activation attack. To mitigate these security risks, we propose a random nonce along with $AppKey$ to randomize the key for each session. The security analysis of the proposed enhancement demonstrates that the adversary cannot execute the attacks with non-negligible probability. As a future work, we plan to implement the proposed scheme in a real environment.

## Acknowledgements

This work is supported by *New Faculty Seed Grant (Project ID: 1548)*, BITS Pilani, India.

## References

[1] "TS001-1.0.4 LoRaWAN L2 1.0.4 Specification," https://resources.lora-alliance.org/technical-specifications/ts001-1-0-4-lorawan-l2-1-0-4-specification, accessed: 2024-01-01.

[2] B. Oniga, V. Dadarlat, E. De Poorter, and A. Munteanu, "Analysis, design and implementation of secure lorawan sensor networks," in *2017 13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*. IEEE, 2017, pp. 421–428.

[3] I. Butun, N. Pereira, and M. Gidlund, "Analysis of lorawan v1. 1 security," in *Proceedings of the 4th ACM MobiHoc Workshop on Experiences with the Design and Implementation of Smart Objects*, 2018, pp. 1–6.

[4] M. Eldefrawy, I. Butun, N. Pereira, and M. Gidlund, "Formal security analysis of lorawan," *Computer Networks*, vol. 148, pp. 328–339, 2019.

[5] C. El Fehri, N. Baccour, P. Berthou, and I. Kammoun, "Experimental analysis of the over-the-air activation procedure in lorawan," in *2021 17th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, 2021, pp. 30–35.

[6] F. Kuntke, V. Romanenko, S. Linsner, E. Steinbrink, and C. Reuter, "Lorawan security issues and mitigation options by the example of agricultural iot scenarios," *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 5, p. e4452, 2022.

[7] D. Naidu and N. K. Ray, "Review on authentication schemes for device security in lorawan," in *2021 19th OITS International Conference on Information Technology (OCIT)*. IEEE, 2021, pp. 387–392.

[8] X. Yang, E. Karampatzakis, C. Doerr, and F. Kuipers, "Security vulnerabilities in lorawan," in *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE, 2018, pp. 129–140.

[9] I. Butun, N. Pereira, and M. Gidlund, "Security risk analysis of lorawan and future directions," *Future Internet*, vol. 11, no. 1, p. 3, 2018.

[10] F. Mårlind and I. Butun, "Activation of lorawan end devices by using public key cryptography," in *2020 4th Cyber Security in Networking Conference (CSNet)*. IEEE, 2020, pp. 1–8.

[11] V. Ribeiro, R. Holanda, A. Ramos, and J. J. Rodrigues, "Enhancing key management in lorawan with permissioned blockchain," *Sensors*, vol. 20, no. 11, p. 3068, 2020.

[12] ——, "A fault-tolerant and secure architecture for key management in lorawan based on permissioned blockchain," *IEEE Access*, vol. 10, pp. 58 722–58 735, 2022.

[13] H. Noura, T. Hatoum, O. Salman, J.-P. Yaacoub, and A. Chehab, "Lorawan security survey: Issues, threats and possible mitigation techniques," *Internet of Things*, vol. 12, p. 100303, 2020.

[14] T. Hess, D. Bol, and R. Sadre, "Ultra-low-power over-the-air-update in secure lorawan networks," *Ecole polytechnique de Louvain, Université catholique de Louvain*, 2020.

[15] H. N. Noura, O. Salman, T. Hatoum, M. Malli, and A. Chehab, "Towards securing lorawan abp communication system." in *CLOSER*, 2020, pp. 440–447.

[16] S. Milani and I. Chatzigiannakis, "Design, analysis, and experimental evaluation of a new secure rejoin mechanism for lorawan using elliptic-curve cryptography," *Journal of Sensor and Actuator Networks*, vol. 10, no. 2, p. 36, 2021.

[17] J. J. Barriga and S. G. Yoo, "Securing end-node to gateway communication in lorawan with a lightweight security protocol," *IEEE Access*, vol. 10, pp. 96 672–96 694, 2022.

[18] B. Orazbayev, "Jamming lorawan over the air authentication," 2022.

[19] R. Fujdiak, K. Mikhaylov, J. Pospisil, A. Povalac, and J. Misurec, "Insights into the issue of deploying a private lorawan," *Sensors*, vol. 22, no. 5, p. 2042, 2022.

[20] R. Sujatha and V. Radovic, "Security enhancement of joint procedure based on improved elliptic curve cryptography in lorawan," *Wireless Personal Communications*, vol. 129, no. 3, pp. 1471–1487, 2023.

[21] S. Abboud and N. Abdoun, "Enhancing lorawan security: An advanced aes-based cryptographic approach," *IEEE Access*, 2023.

[22] J. Dave, P. Faruki, V. Laxmi, B. Bezawada, and M. Gaur, "Secure and efficient proof of ownership for deduplicated cloud storage," in *Proceedings of the 10th International Conference on Security of Information and Networks*, 2017, pp. 19–26.

[23] J. Dave, A. Dutta, P. Faruki, V. Laxmi, and M. S. Gaur, "Secure proof of ownership using merkle tree for deduplicated storage," *Automatic Control and Computer Sciences*, vol. 54, pp. 358–370, 2020.

[24] J. Dave, S. Saharan, P. Faruki, V. Laxmi, and M. S. Gaur, "Secure random encryption for deduplicated storage," in *Information Systems Security: 13th International Conference, ICISS 2017, Mumbai, India, December 16-20, 2017, Proceedings 13*. Springer, 2017, pp. 164–176.

[25] J. Dave, P. Faruki, V. Laxmi, A. Zemmari, M. Gaur, and M. Conti, "Spark: Secure pseudorandom key-based encryption for deduplicated storage," *Computer Communications*, vol. 154, pp. 148–159, 2020.

[26] J. Dave, N. Choudhury, U. Tiwari, S. Kamtam, and K. S. Rohith, "Secure deduplication with dynamic key management in fog enabled internet of things," in *2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 2023, pp. 1237–1242.