# Extending SOC Capabilities to LoRaWAN: A Cloud-Integrated Intrusion Detection Framework for IoT Networks

**Zsolt Bringye[1], Rita Fleiner[1], Balázs Umhauser[1], Márk Aradi[1], Eszter Kail[1,2]**

[1]Obuda University, Bécsi út 96/b, 1034 Budapest, Hungary

[2] HUN-REN Institute for Computer Science and Control (HUN-REN SZTAKI), Hungarian Research Network, Kende utca 13-17, H-1111 Budapest, Hungary

bringye.zsolt@nik.uni-obuda.hu, fleiner.rita@nik.uni-obuda.hu, w7ysen@stud.uni-obuda.hu, mark@rowra.org, kail.eszter@nik.uni-obuda.hu

*Abstract: The rapid expansion of LoRaWAN-based IoT deployments in critical infrastructure, industrial environments, and smart city applications introduces novel cybersecurity challenges that traditional IT security architectures are neither optimized for nor capable of fully addressing. In this paper, we present a stateful, protocol-aware intrusion detection architecture specifically tailored for LoRaWAN communication, with seamless integration into existing Security Operations Center (SOC) frameworks. Our approach identifies key monitoring points along the communication chain and implements a multi-stage telemetry pipeline that supports protocol-level analysis – even for encrypted traffic. The system successfully detects complex attack scenarios, including man-in-the-middle payload tampering and decryption attacks, validating the framework's effectiveness in practical conditions. The proposed methodology bridges a crucial gap between IoT-specific anomaly detection and enterprise-grade SOC capabilities, offering a scalable and transferable solution for extending cybersecurity visibility into low-power, protocol-constrained networks. Our results highlight both the feasibility and the strategic importance of integrating LoRaWAN telemetry into unified threat detection pipelines.*

*Keywords: IoT; vulnerability analysis; anomaly detection; SIEM; LoRaWAN security*

# 1 Introduction

The rapid proliferation of Internet of Things (IoT) networks across critical infrastructure, industrial automation, smart cities, and healthcare [1] has significantly expanded the cyberthreat surface. Unlike traditional IT environments, IoT networks are composed of resource-constrained and often unattended devices

that communicate over lightweight, application-specific protocols such as LoRaWAN. Their wireless and distributed nature introduces unique challenges in securing them against increasingly sophisticated cyberattacks.

Conventional security mechanisms such as firewalls, intrusion detection systems (IDS), and endpoint protection solutions remain essential but are no longer sufficient in isolation. Especially in low-power, lossy environments, IDS systems must rely on metadata-driven methods due to limited access to packet content or computational resources. While many lightweight IoT anomaly detectors exist, they rarely support SOC-level integration or enable cross-layer event correlation.

To bridge this critical gap, we built a dedicated LoRaWAN test environment, demonstrated in Fig. 1, within our university's infrastructure, leveraging the OTC (Open Telekom Cloud) cloud platform to simulate real-world deployment scenarios. This novel testbed enabled us to rigorously explore the feasibility of extending stateful, multi-layer monitoring into the LoRaWAN ecosystem, not only detecting generic anomalies, but also decoding and analyzing even encrypted or protocol-level interactions indicative of sophisticated IoT-specific threats by correlating metadata across the LoRaWAN pipeline.

Our work represents a strategic shift in IoT cybersecurity, moving beyond lightweight anomaly detectors limited to edge nodes. Instead, we propose a unified, SOC-integrated intrusion detection system (IDS) that identifies and correlates threat signals from multiples sources, across the entire LoRaWAN communication pipeline – from end-devices to gateways to network servers. By pinpointing key monitoring junctures in the communication flow, we were able to implement targeted capture mechanisms, which feed into a pre-configured, ELK-based SIEM environment. This allows real-time, structured, and context-aware alerting within a familiar SOC interface.

In this article, we outline the technical underpinnings of our system and detail how our pipeline enables deep, protocol-specific analysis. We also demonstrate the framework's efficacy through the detection of real-time man-in-the-middle (MITM) attacks on LoRaWAN traffic, including payload manipulation and decryption, thereby validating the practicality of our method in a cloud-hosted SOC context.

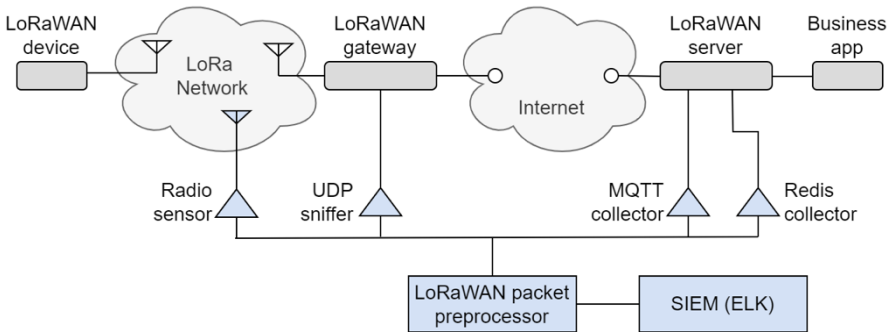


Figure 1
Main components of LoRaWAN Test Environment at Obuda University

The key contributions of our research include the following:

- Design and deployment of a LoRaWAN testbed with cloud-native infrastructure support: We established a fully functional, university-hosted LoRaWAN test environment in conjunction with the OTC cloud platform. This infrastructure enabled controlled experimentation with real traffic flows and realistic attack scenarios, serving as a foundation for validating IDS performance in a production-like setting.
- A stateful, multi-point monitoring architecture for LoRaWAN networks: By identifying and instrumenting critical points across the communication chain (e.g., end-devices, gateways, and network servers), we implemented a fine-grained monitoring pipeline that allows deep inspection and event correlation even for encrypted or protocol-obscured traffic patterns.
- Seamless integration into existing SOC environments using ELK-based SIEM tools: Our system is designed for compatibility with enterprise-grade SOC operations. Through the use of ELK stack components, we provide an extensible and maintainable way to define LoRaWAN-specific alert rules, visualize telemetry data, and cross-reference IoT events with conventional IT security logs.
- Demonstrated effectiveness in detecting real-world attack scenarios: We validated the practical value of the proposed system by simulating two types of man-in-the-middle (MITM) attacks: a payload modification and a denial-of-service attempt, and successfully generating actionable alerts within the SIEM. These scenarios illustrate the framework's ability to surface subtle, protocol-level anomalies that are typically invisible to generic IDS tools.
- Blueprint for extending SOC visibility into IoT ecosystems: Beyond the technical implementation, our work presents a transferable architecture for organizations aiming to extend their existing security monitoring infrastructure to cover IoT deployments. The framework provides a scalable, adaptable model for incorporating low-power, low-data-rate IoT networks into broader threat detection and response strategies.

The structure of the paper is as follows: Section 2 provides an overview of related work on anomaly detection and intrusion detection, with a focus on both traditional and IoT-specific approaches. In Section 3 a brief overview of the LoRaWAN protocol and operation is presented. Section 4 outlines the experimental setup and Section 5 presents the attack scenarios and detection results. Finally, the Conclusion summarizes our work and highlights possible future directions.

# 2 Related Work

Anomaly detection methods in LoRaWAN differ significantly from those used in traditional networks, due to fundamental differences in architecture, energy constraints, and communication characteristics. LoRaWAN devices are typically low-power and resource-constrained, which makes the use of lightweight, energy-efficient detection techniques essential. Moreover, LoRaWAN's long-range and low-bandwidth communication model results in frequent packet loss and limited data rates, further complicating the application of traditional anomaly detection approaches. In contrast, conventional networks benefit from high data throughput and ample computational capacity, enabling the deployment of resource-intensive methods such as deep learning and advanced traffic analysis. These disparities render many traditional detection techniques ineffective in LoRaWAN environments, prompting the development of specialized anomaly detection strategies tailored to low-power, high-latency, and lossy communication contexts.

In response to these challenges, a growing body of research has focused on developing tailored anomaly and intrusion detection solutions for LoRaWAN networks. The following paragraphs review relevant literature in this domain.

IoT networks characterized by heterogeneous systems and often changing environments has led to new attack vectors, thus enabling artifcial intelligence based tools to serve promising detection techniques. Also, the emergence of Egde and fog computing in IoT networks has introduced new IDS or anomaly detection methods.

Despite these promising approaches, several studies rely heavily on synthetic datasets or simulated environments, which may not reflect the complexities of real-world LoRaWAN deployments. Esteves et al. [2] highlight that many existing intrusion detection methods lack validation against realistic traffic and fail to address deployment challenges in constrained environments. Their work proposes a lightweight, edge-native intrusion detection framework specifically tailored for LoRaWAN, capable of analyzing traffic directly at the gateway level without requiring centralized processing. Their system combines signature-based rules (via Suricata) with behavioral anomaly detection using a K-Nearest Neighbors classifier and analyzes packet metadata (RSSI, SNR, payload size, SF, etc.) to detect intrusions without inspecting encrypted payloads, making it privacy-preserving and

efficient. Designed for deployment in both centralized and distributed (edge) scenarios, the IDS demonstrates high accuracy in detecting anomalies and malicious behavior, including replay attacks, misconfigurations, and potential network disruptions. The approach is validated on real-world data and adapted to dynamic environments like mobile gateways in railways and search-and-rescue missions. The authors emphasize the importance of real-world evaluation and demonstrate the effectiveness of their solution using actual LoRaWAN gateway traffic. Similarly, Milani et al. [3] underline the necessity of deploying intrusion detection as close to the data source as possible, preferably at the edge, to improve responsiveness and reduce backhaul overhead. They introduced EdgeLora, an architecture enabling on-gateway data processing to reduce latency and bandwidth usage while preserving protocol security and scalability. Spadaccino et al. [4] also support this perspective, noting that edge-based IDS solutions, by operating near the data source, can deliver faster detection and reaction times—especially important in time-critical IoT scenarios.

Proto et al. [5] propose LADE, a lightweight intrusion detection architecture for LoRaWAN sensors that identifies energy depletion attacks (EDAs) based on local energy consumption analysis. Their system deploys both learning and detection modules directly on constrained IoT devices, using statistical distance metrics (Sibson divergence) to compare real-time energy profiles. LADE effectively detects both active jamming and silent firmware-level attacks that drain batteries without generating network traffic. While LADE does not monitor protocol-level anomalies (e.g., MIC tampering or join message abuse), it uniquely enables autonomous detection of hardware- and firmware-induced EDAs without reliance on network traffic analysis.

Authors in [6] presents LoRaLOFT, a novel intrusion detection method on MAC layer in LoRaWAN networks. The study targets two major attack types; greedy behavior where compromised devices selfishly violate MAC constraints like duty cyle and an attack behavior, where malicious nodes intentionally flood the network with traffic to disrupt legitimate communication. LoRaLOFT, combines a rule-based threshold system with an unsupervised machine learning model – Local Outlier Factor (LOF) with a dyanmic detection metric – the number of packet, or energy consumption to detect unexpected behaviour. The authors managed to detect the above-mentioned anomalies with high accuracy and low false-positive rates. This solution enables the identification of malicious nodes behaviour using computationally lightweight metrics, eliminating the need for deep packet inspection or centralized analysis, and thereby enabling deployment in resource-constrained environments such as gateways or fog-level processing nodes.

Babazadeh et al. in [7] also proposes a LoRa-based anomaly detection framework that operates on both the sensor (edge) and center (cloud) sides, aiming to minimize data transmission and energy consumption while maintaining high detection reliability. Unlike traditional IDS approaches, this system does not rely on machine learning or traffic analysis but instead uses data compressibility as an indicator of

anomalous behavior. Each sensor locally monitors its collected data and calculates a compression rate; significant drops in compressibility signal a potential anomaly. Only the most suspicious event in each cycle is reported to the center via lightweight alert messages. Upon receiving an alert, the central node requests and reconstructs the compressed data for in-depth analysis. This method is optimized for low-power, bandwidth-constrained LoRa environments, offering an efficient alternative for anomaly detection without needing continuous payload transmission or complex computation on the sensor.

Kurniawan and Kyas in their work [8] also base their detection methods on metadata. They present a machine learning-based anomaly detection system tailored for LoRaWAN gateways. Their approach focuses on monitoring communication metadata – such as RSSI, SNR, and packet timing – to detect potential anomalies without analyzing encrypted payloads. By collecting real-world LoRaWAN traffic data and evaluating it using eleven different outlier detection algorithms (e.g., LOF, KNN, CBLOF, PCA), they demonstrate that lightweight anomaly detection is feasible even on constrained devices like Raspberry Pi. The study shows promising accuracy and performance across multiple anomaly types, including replay attacks and flooding, making it a strong candidate for gateway-level intrusion monitoring.

Table 1
Cryptographic Keys Used for Message Protection in LoRaWAN 1.0.3

| Study | Methodology | Monitoring location | Evaluation | Key findings | Limitation |
|---|---|---|---|---|---|
| [2] | KNN-based anomaly detection | Gateway (edge) and centralized SOC | Quantitative (real network data, >90% accuracy) | Edge-based IDS using real network traffic | Focused on statistical traffic anomalies, not protocol logic |
| [3] | Group-key encryption and local processing (Edge2LoRa) | Gateway and edge | Experimental demo | Reduces latency and network load | No intrusion/anomaly detection logic |
| [4] | Conceptual comparison | Edge/cloud | Conceptual | Highlights need for lightweight ML at edge | No implementation for LoRaWAN |
| [5] | Sensor side energy consumption pattern based | Sensor level | Quantitative (F1 ≈ 0.93) | Detects silent energy depletion attacks | Limited to Energy Depletion Attacks |
| [6] | Threshold-based | Network server | Simulation | Detects greedy/flooding behavior | Focuses on MAC layer anomalies |
| [7] | Data compressibility- | Sensor level | Prototype with alerts | energy-efficient local filtering | Anomaly detection at sensor side |

| | | | | | |
|---|---|---|---|---|---|
| | based anomaly detection | | | | |
| [8] | Metadata-based anomaly detection | Gateway level | Quantitative | detection is feasible on constrained devices | No protocol state modelling |

To summarize, while the reviewed literature offers a diverse set of anomaly detection approaches for LoRaWAN, most solutions remain limited either to localized heuristics or operate without explicit protocol-awareness. Notably, none of these studies model protocol states formally or support cross-layer, SOC-integrated monitoring.

To clearly illustrate these differences, Table 1 presents a qualitative comparison of the most relevant LoRaWAN IDS approaches discussed above, detailing their methodology, monitoring location, evaluation type, along with key findings and limitations compared to our proposed solution. As shown, our work is, to the best of our knowledge, the first to combine both IoT and legacy network environments into SOC-integrated intrusion detection framework with multi-point protocol-level telemetry enabling anomaly detection based on well-defined state transitions in the LoRaWAN protocol.

# 3 LoRaWAN Communication

## 3.1 Overview

LoRaWAN (Long Range Wide Area Network) [9] is a low-power, wide-area networking protocol designed for wireless communication with IoT devices. It operates on top of LoRa (Long Range), a chirp spread spectrum modulation technique, which is ideal for IoT devices that require extensive coverage and prolonged battery life.

LoRaWAN defines the communication protocol and system architecture for the network. It specifies how devices communicate with gateways, which then connect to network servers. The architecture includes **End Devices** with sensors or actuators, equipped with LoRa modules to collect information from its surroundings, **Gateways** to relay messages between end devices and network servers and servers:

**Network Server** for managing the network, handling MAC layer operations, and performing message deduplication. They also manage the routing of the messages, keeps track of the devices in the network, and ensures secure communication.

**Join Server** primarily responsible for device authentication, session key generation, distribution and storage. It is a newly introduced architecture element in LoRaWAN 1.1.

**Application Servers** for processing application data and providing interfaces for end-users.

LoRaWAN has evolved through several specification versions, with LoRaWAN 1.0.x forming the foundation for basic functionality and wide adoption, while LoRaWAN 1.1 [10] introduced significant enhancements such as improved roaming support, finer-grained security mechanisms, and the formal separation of the Join Server. These developments reflect the protocol's maturation toward supporting more robust and scalable IoT deployments.

### 3.1.1 Device Classes

LoRaWAN defines three classes of end devices: Class A, B and C.

**Class A**: Provides bidirectional communication. It is the most energy-efficient class, where devices open two short receive windows after transmitting uplink data. This class is suitable for applications where uplink communication predominates, and downlink messages are not time-critical.

**Class B**: Adds scheduled receive windows, allowing devices to receive downlink messages at predetermined times. The support for Class B devices were introudced in LoRaWAN 1.0.3.

**Class C**: Keeps receive windows open almost continuously, except when transmitting, making it ideal for applications requiring frequent downlink communication.

According to the LoRaWAN standard [9], all implemented modules are required to adhere to Class A specifications. Classes B and C are optional and may be used for specific design scenarios, to accommodate different application needs.

### 3.1.2 LoRaWAN Packet Structure, and Message Types

In the LoRaWAN protocol, the structure of the payload depends on the purpose of the message, which may be a Join Request (used during Over-The-Air Activation – OTAA), Join Accept (response from the network to a Join Request), or a Data Frame (used for application-level data and MAC commands). The structure of the payload in the case of a Data Frame is depicted in Fig. 2, which illustrates the internal composition of a LoRaWAN Data Frame. The payload consists of a MAC header (MHDR), a MAC payload – which includes the Frame Header (FHDR), an optional FPort field, and the actual FRMPayload – and finally the Message Integrity Code (MIC). The Frame Header itself contains addressing and control information such as the DevAddr, Frame Control (FCtrl), Frame Counter (FCnt), and optional MAC commands. The FRMPayload may contain either application data or MAC

commands, depending on the value of the FPort. This layered structure enables both application communication and control signaling within a single, secure frame.

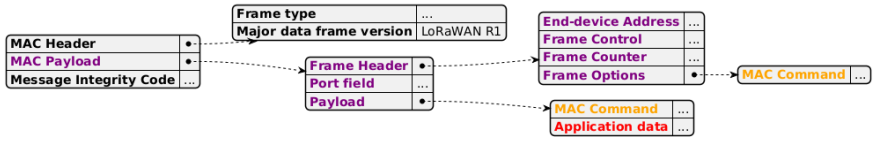


Figure 2
Structure of the payload of a Data Frame

LoRa radio packets can follow either an explicit or implicit format. In this work, we focus on the explicit format, as it is used in the communication scenarios we examined. The key difference between uplink and downlink transmissions is that downlink packets omit the trailing CRC (Cyclic Redundancy Check) field in order to reduce channel occupancy.

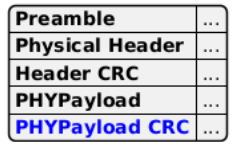


Figure 3
Structure of the LoRa physical layer frame

Fig. 3 shows the format of the LoRa physical layer frame. The application-relevant data mandated by the LoRaWAN protocol resides in the `PHYPayload` field, whose maximum size depends on the regional parameters.

## 3.2 Device Activation and Key Management in LoRaWAN

Secure communication is a fundamental requirement in LoRaWAN-based IoT networks, where devices often operate unattended in untrusted environments. To ensure both confidentiality and integrity of data, LoRaWAN employs symmetric cryptographic mechanisms based on session keys. These keys are used to encrypt the application payload and to generate Message Integrity Codes (MICs), which authenticate the origin and content of each message.

The Message Integrity Code (MIC) is used to verify the authenticity and integrity of the message. The MIC is calculated based on specific fields of the message and is appended to the end of the PHYPayload. It allows the receiving network components to detect any unauthorized modifications to the message during transmission. The exact fields included in the MIC calculation depend on the message type (e.g., Join Request, Data Frame, etc.).

### 3.1.3 Device Activation in LoRaWAN

The LoRaWAN specification defines two methods for activating end devices: Over-the-Air Activation (OTAA) and Activation by Personalization (ABP).

The details of the join procedure differ from version to version. In our implementation, we focus on the LoRaWAN 1.0.3 join procedure because this is the version currently used in our testbed environment. Despite the release of newer specifications such as LoRaWAN 1.1, version 1.0.3 remains widely adopted in many real-world production deployments due to its relative simplicity, compatibility with legacy devices, and maturity in terms of vendor support. Importantly, the widely used open-source ChirpStack network server, which serves as the backbone for numerous academic and industrial LoRaWAN installations, continues to use 1.0.3 as its default implementation target. As a result, analyzing version 1.0.3 offers both practical relevance and technical clarity for real-world applications. The following section describes the procedure based on the LoRaWAN 1.0.3 [9] specification, which is a slightly simpler method for generating session keys and uses fewer keys to secure communication than in specification 1.1 [10].

In OTAA, which is the preferred and more secure method, devices perform a join procedure at the beginning of their operation. The end device sends a Join Request message containing its DevEUI, AppEUI, and a randomly generated DevNonce. The Network Server responds with a Join Accept message that includes a JoinNonce (a server-generated random number), NetID, a dynamic device address (DevAddr), and other parameters such as RXDelay and DLSettings. Both the Join Request and Join Accept messages are protected using a pre-shared AppKey, which is known to both the end device and the network server.

Using the AppKey and values from the join exchange (DevNonce, JoinNonce), the end device derives two session keys: The NwkSKey (Network Session Key), used to ensure message integrity and authenticate MAC-level communication, and the AppSKey (Application Session Key), used to encrypt and decrypt the application payload. These session keys are unique and valid per session and ensure confidentiality and authenticity of LoRaWAN communications in dynamic environments.

In contrast, ABP assigns the DevAddr, NwkSKey, and AppSKey directly to the end device during provisioning. No join exchange is performed during runtime. While ABP simplifies deployment, especially in networks with no reliable downlink, it provides lower security. The keys remain static and may be reused across devices or sessions, making the system more vulnerable to key compromise or cloning.

### 3.1.4 Key Usage in Message Protection

Without access to the AppKey and the Join procedure, session keys cannot be reconstructed. Consequently, neither the application payload nor the MIC can be

properly decrypted, validated, or regenerated. This limitation prevents attackers or analysts without access to keys from meaningfully modifying or verifying LoRaWAN messages. Table 1 summarizes the cryptographic keys used for encrypting and validating messages, depending on the message type.

Table 1
Cryptographic Keys Used for Message Protection in LoRaWAN 1.0.3

| Msg type | Encryption key | Msg integrity key |
|---|---|---|
| MACPayload | NwkSKey[1] (MAC commands) | NwkSKey |
| MACPayload | AppSKey | NwkSKey |
| Join-Request | no encryption | AppKey |
| Join-Accept | AppKey | AppKey |

[1]In case of MAC commands, however, MAC commands can also be transmitted in the unencrypted FOpts field

Without access to the AppKey and the Join procedure, session keys cannot be derived, and neither the application payload nor the MIC can be properly interpreted or regenerated. Table 2 outlines what can or cannot be accessed or modified without the necessary keys.

Table 2
Decryption and modification possibilities without keys

| Data type | Readable without key | Modifiable without detection |
|---|---|---|
| **Payload** | No | No |
| **MIC** | Partially* | No |

*Only metadata-level observables; integrity verification not possible

### 3.1.5 Data Enrichment in LoRaWAN Communication

In addition to the payload data directly transmitted by end devices, LoRaWAN communication is enriched with multiple layers of metadata during transmission. At the physical (radio) layer, the gateway captures additional contextual information such as RSSI, SNR, frequency, channel, and timestamp, which are appended to each received RF packet and may be leveraged for anomaly detection or forensic analysis.

Fig. 4 illustrates the key metadata fields and gateway status parameters associated with each received packet. Beyond the base64-encoded payload, the gateway contributes radio-level attributes and periodically broadcasts its own status, including GPS coordinates, system time, and statistics on packet reception, forwarding, and emission. These enriched data fields serve as valuable inputs for protocol-level monitoring and detection of suspicious behavior.

Moreover, network servers, such as ChirpStack, can further enhance the data context by linking messages to registered device names, identifiers, and session state. If the Join procedure can be observed and the AppKey is known (or discoverable), session keys can be derived, allowing decryption of the payload and deeper inspection of the application-layer data. This multi-layer enrichment process significantly expands the analytical potential of LoRaWAN communication and provides a rich foundation for cross-layer anomaly detection strategies.

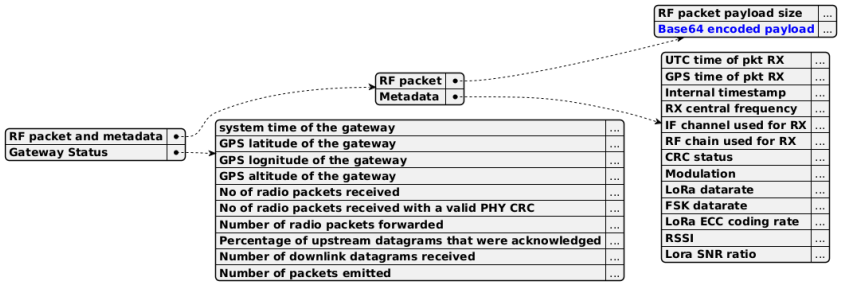


Figure 4

Metadata and status information added by the LoRaWAN gateway to each RF packet

# 4 The LoRaWAN Security Testing Environment

The constructed test environment consists of a physical LoRaWAN network operated locally at our institute, complemented by additional components deployed in a cloud infrastructure to support traffic observation and data collection.

## 4.1 Elements of Our LoRaWAN Network

The core of the LoRaWAN setup includes two RAK Wireless gateways (RAK7246G WisGate Developer D0) and an open-source ChirpStack [11] LoRaWAN server hosted in the T-Systems OTC public cloud. Both gateways are implemented as Raspberry Pi HAT modules and thus require Raspberry Pi hardware to operate.

The first gateway is based on a Raspberry Pi 4 and connects directly to the internet via the institutional firewall, forwarding packets to the ChirpStack server via MQTT. The second gateway uses a Pi Zero 2 and communicates locally with a Raspberry Pi 3 running the ChirpStack MQTT Forwarder. This forwarder receives UDP-formatted packets from the gateway and relays them to the cloud-based ChirpStack server using MQTT. This dual-gateway architecture enables the parallel monitoring of both MQTT and UDP-based communication paths.

The virtual machines running the ChirpStack server and the ELK stack (Elasticsearch, Logstash, Kibana) are deployed within the T-Systems OTC cloud. The MQTT broker (Eclipse Mosquitto), along with PostgreSQL and Redis services required by ChirpStack, are also hosted in this cloud environment. The structure of our test environment is depicted in Fig. 5.

Periodic data transmission is provided by COST sensors (Seeed Studio SenseCAP S2101 and S2102), which transmit measured values over the LoRaWAN network. The S2101 sensor sends ambient temperature and humidity, while the S2102 transmits light intensity. Additionally, both sensors include the battery level every 20 measurements.

The environment also includes Seed Studio Wio-E5 mini LoRaWAN clients connected to Raspberry Pi boards via serial interface. Controlled via AT commands, these devices support a variety of test scenarios, including invalid keys and unregistered device attempts. However, the AT interface does not support sending protocol-level malformed packets (e.g., with CRC errors), as such functionality is not exposed by the command set.

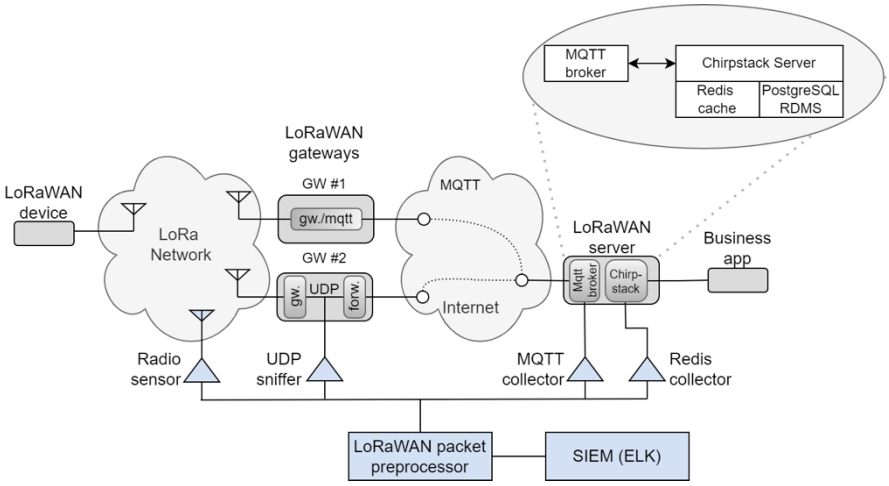


Figure 5

Overview of LoRaWAN test environment, showing monitoring points, indicated with blue triangles across the LoRaWAN communication pipeline. Uplink messages from LoRaWAN end devices are received by gateways (GW#1, GW#2) and forwarded to the ChirpStack network server. The ChirpStack stack includes components such as Redis, PostgreSQL, and an MQTT broker for handling telemetry and device data. All collected data is processed by the LoRaWAN packet preprocessor and forwarded to the SIEM for correlation, visualization, and real-time alerting.

## 4.2 Monitoring and Collection of LoRaWAN Communication

In order to create a comprehensive IDS (Intrusion Detection System) implementation, we aimed to monitor LoRaWAN communication at every possible point where modification or interference could potentially occur. Based on an analysis of the communication pipeline, we identified five distinct locations where traffic can be captured and analyzed.

LoRaWAN communication is observed and collected through five different methods, two of which have already been implemented and tested beyond proof-of-concept (PoC) level. The remaining three are currently operational in PoC form. Table 3 summarizes the different monitoring possibilities in our test environment.The monitoring solutions are as follows:

Software-defined radio (SDR)-based signal reception, with fully software-based decoding and analysis of raw LoRa signals.

Reception and analysis using two RAK7246G WisGate Developer D0 Gateways and custom Raspberry Pi-based software. A similar solution was described in [12], and our system is functional at the PoC level. However, further configuration is still required for the Semtech SX1308 radios. The implementation was based on Semtech's publicly available sample code [13].

Interception of UDP traffic between the LoRaWAN gateway and the MQTT forwarder, using widely adopted packet sniffing tools [14]. This method is still in the PoC phase, and integration with the ELK stack for data forwarding has yet to be completed.

Monitoring MQTT communication between the gateway's MQTT forwarder and the ChirpStack server. By subscribing to the appropriate MQTT topics, LoRaWAN packet data can be captured without interfering with live system operation. In production environments, these MQTT messages are typically encoded using Protobuf. The Protobuf schema is available from the ChirpStack GitHub repository [10], [11], which allows custom decoders to be developed. Decoded messages are transformed into JSON format and forwarded to the ELK stack.

Accessing dynamic data stored by ChirpStack in Redis, such as LoRaWAN frame logs. Using the appropriate API key (which can be generated from the ChirpStack admin interface), these Protobuf-formatted messages can be retrieved and decoded into JSON. The processed messages are then forwarded to the ELK stack for further analysis.

## 4.3 SIEM-Based Telemetry Pipeline for LoRaWAN Security Monitoring

To support structured monitoring of LoRaWAN communications, we implemented a Security Information and Event Management (SIEM) pipeline based on the open-source ELK stack (Elasticsearch, Logstash, Kibana) [15]. The pipeline is designed

to ingest, process, enrich, and store LoRaWAN traffic data from multiple sources, enabling real-time visualization, correlation, and anomaly detection within a SOC environment. The system relies on a LoraWAN packet preprocessor with two independent middleware components, each responsible for processing LoRaWAN messages from distinct sources:

Table 3
Comparison of the monitoring approaches

| Method | Readiness | Required Access Level |
|---|---|---|
| Radio signal capture (SDR) | PoC | Physical proximity within the LoRa radio range (potentially up to several hundred meters) |
| Radio signal capture (LoRa radio) | PoC | Physical proximity within the LoRa radio range (potentially up to several hundred meters) |
| UDP traffic interception | PoC | Access to the local network (LAN). If using Wi-Fi, no physical connection is needed, but joining the network is required |
| MQTT packet monitoring | Ready | Network access and connection to the MQTT broker; depending on its security configuration, this can range from simple (no security) to complex (certificate-based access) |
| Redis data access | Ready | Network access and a valid Redis API key, which requires ChirpStack administrative privileges |

**Frame Log Middleware – Redis-Based Source**

The first component retrieves LoRaWAN frame logs directly from the ChirpStack network server's Redis stream, decoding Protobuf [16] messages into JSON using ChirpStack's API bindings, and enriching the entries with contextual metadata such as device name and profile. This enrichment is achieved through a preloaded in-memory dictionary generated via ChirpStack's API calls. Newly detected devices are dynamically registered into this dictionary based on their Join Request data.

**Payload Decoder Middleware – MQTT-Based Source**

The second middleware supports payload decryption, where available, through an external decoder module [17] integrated into the workflow. The decoder derives session keys (AppSKey and NwkSKey) by using stored AppKeys, retrieved from the ChirpStack database, and by observing Join Request and Join Accept messages.

Although payloads and MICs are inaccessible without keys, our system leverages observable non-payload fields, such as frame type, DevEUI, DevNonce, timing metadata, RSSI, and SNR, to detect anomalies even in encrypted traffic. Behavioral inconsistencies like repeated join attempts, DevNonce reuse, invalid message

sequences, denial-of-service conditions, or other protocol misuse patterns can be identified through protocol-conformant state tracking and metadata correlation. When decryption succeeds, the application-layer payload is parsed into structured JSON and forwarded to Logstash.

# 5 Case Study: Application of the Monitoring Framework

To evaluate the effectiveness of our proposed multi-layer detection architecture, we implemented a real-world man-in-the-middle (MITM) attack targeting the LoRaWAN communication chain between the gateway and the ChirpStack network server. The goal of the attack was twofold: (i) to manipulate location-related metadata (specifically, GPS coordinates) in transit, and (ii) to escalate the attack by decrypting encrypted payloads through session key extraction.

## 5.1 GPS Location Manipulation Attack and Detection

In the first phase, we intercepted and altered live LoRaWAN traffic using ARP spoofing to redirect UDP packets (port 1700) through a Kali Linux-based attacker node. The captured packets were parsed using tshark, transformed to JSON format, and manipulated in real-time using a Python script built on the scapy library. The script identified LoRaWAN packets and replaced legitimate gateway GPS metadata (latitude, longitude, altitude) with randomly generated coordinates before forwarding the modified packets to the original destination (ChirpStack). This manipulation was repeated periodically every 15 minutes to simulate a mobile gateway scenario, creating misleading geolocation data for the network.

The overall architecture of the interception and manipulation process, including the attack point and data flow through the monitoring pipeline, is illustrated in Fig. 6.

To detect spoofing-based manipulation of GPS metadata in LoRaWAN communication, we defined a custom SIEM rule within the Kibana interface. This rule leverages our gps.loc enrichment field, created during preprocessing, which captures the geolocation metadata associated with each received packet. Assuming that legitimate devices remain geographically static, the system groups incoming messages by devName and counts the number of distinct coordinate values observed over a short time window. Any unexpected change in the unique gps.loc value is treated as anomaly and triggers an alert. This threshold-based approach was implemented using Kibana's native detection engine and proved effective in flagging anomalous mobility patterns caused by GPS spoofing.

To illustrate the impact of such GPS spoofing attacks, we visualized the manipulated gateway coordinates on a geolocation map in Kibana. Leveraging the

custom gps.loc field in our enriched logs, we configured a clustered map visualization that aggregates device occurrences based on geographic location.

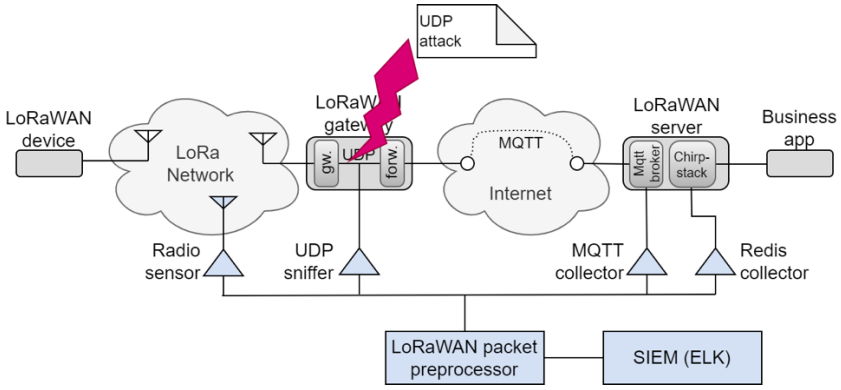


Figure 6

LoRaWAN security monitoring pipeline highlighting the UDP interception point between the gateway and the network server. The red lightning symbol represents the point where man-in-the-middle attacks are executed in our scenarios.

This allowed us to clearly observe artificial location shifts over time, reflecting the simulated mobility caused by the cyclic injection of falsified coordinates. The resulting visualization in Fig. 7 demonstrates the dynamic and inconsistent geographic distribution of the gateway, which can serve as a strong indicator of suspicious behavior in a real-world monitoring context.

The duplicated appearance of each manipulated location in the visualization originates from the data processing architecture, where two independent middleware components ingest overlapping sets of LoRaWAN frame logs from different sources. One middleware captures frame data from the Redis-based ChirpStack stream, while the other processes decoded payloads received via the MQTT broker. As both pipelines forward enriched entries to the SIEM system independently, the same logical transmission event may appear twice in the final dataset. This effect leads to duplicated GPS coordinates in the Kibana map, even though only a single gateway transmission occurred in the physical environment.

## 5.2 Payload Decryption

In the second phase, we extended the attack to decrypt uplink payloads by exploiting LoRaWAN session key derivation. Using the Loracrack toolset [17] developed by Applied Risk, the attacker captured Join Request and Join Accept messages and performed a dictionary-based attack to retrieve the AppKey. Provided that the key was known or weak, session keys were derived and subsequently used to decrypt encrypted Unconfirmed Data Up or Confirmed Data Up payloads.

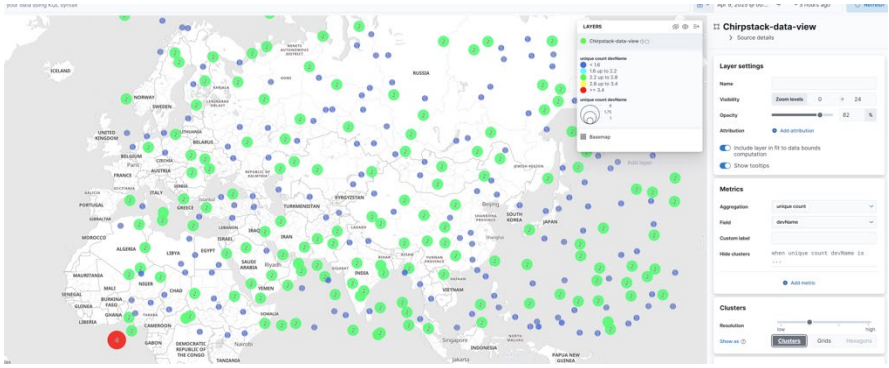


Figure 7

Clustered map visualization of manipulated gateway GPS positions in Kibana

The map shows LoRaWAN gateway positions reported in enriched logs under simulated GPS spoofing conditions. Each green dot represents a unique falsified location, while the number inside the dot indicates how many times that specific coordinate appeared in the telemetry logs

The attack was successfully executed in a mixed physical and virtual environment. It demonstrated that under certain conditions, (e.g., weak or known AppKeys) an adversary cannot only inject falsified metadata but also gain access to sensitive sensor information transmitted within encrypted LoRaWAN payloads. Our system was able to detect both aspects of the attack: the geolocation manipulation (via payload field validation and cross-layer inconsistency detection), and the abnormal session behavior (via MIC validation and correlation of Join messages within the SOC environment). Fig. 8 shows how decoded environmental sensor data from a LoRaWAN device is displayed in the SIEM, including temperature, humidity, $CO_2$, etc.

| devAddr | mType | origin | decoded.Battery | decoded.StationId | decoded.Pressure | decoded.Temperature | decoded.Lux | decoded.Intensity | decoded.Humidity | decoded.CO2 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0032c82b | UNCONFIRMED_DATA_UP | redis_stream | 49% | 48 | 5.0 Kpa | 4.8 Celsius | 4.8 Lux | 5.7% | 4.8% | 51 ppm |
| 0032c82b | UNCONFIRMED_DATA_UP | redis_stream | 49% | 48 | 5.0 Kpa | 4.8 Celsius | 4.8 Lux | 5.7% | 4.8% | 51 ppm |

Figure 8

Decoded sensor data displayed in the SIEM system

## Conclusions

In this paper, we presented a practical and scalable approach for integrating LoRaWAN-based anomaly detection into centralized SOC environments. By establishing a full-featured testbed built on OTC cloud infrastructure and focusing on the widely adopted LoRaWAN 1.0.3 specification, we demonstrated how multi-layer monitoring can be applied across the LoRaWAN communication pipeline. Our system identifies strategic observation points, along the communication pipeline, where protocol-aware telemetry was used to feed meaningful data into an

ELK-based SIEM platform. We showed that even without decrypting payloads, valuable insights can be extracted from metadata and message structure, enabling real-time detection of critical attack scenarios such as payload manipulation and denial-of-service. Moreover, our framework bridges the visibility gap between traditional IT infrastructure and low-power, IoT-specific communication, laying the groundwork for unified security monitoring.

While the current study focuses on a small-scale testbed and a specific LoRaWAN version (1.0.3), this choice reflects the version's widespread use in production systems, particularly in Central Europe, and allows for reproducible experimentation in a controlled environment. Despite this scope, the proposed framework is not tightly coupled to a specific protocol version. On the contrary, its modular design and emphasis on protocol-conformant telemetry make it highly adaptable to newer LoRaWAN specifications or even to other IoT communication protocols. Therefore, we consider our implementation a blueprint for extending SOC-based visibility into constrained IoT networks. Future work will include broader protocol coverage, real-world deployment at larger scale, and the integration of advanced machine learning-based detection capabilities, which together will further enhance the effectiveness and applicability of our approach in securing next-generation IoT environments.

## Acknowledgement

This research was supported with access to the Open Telekom Cloud, provided by Deutsche Telekom TSI Hungary Ltd. We gratefully acknowledge the infrastructure support, which contributed to the results presented in this publication and helped promote the T-Systems Open Telekom Cloud both nationally and internationally.

## References

[1] T. E. Ali, F. I. Ali, P. Dakić, and A. D. Zoltan, “Trends, prospects, challenges, and security in the healthcare internet of things,” *Computing*, Vol. 107, No. 1, p. 28, Dec. 2024, doi: 10.1007/s00607-024-01352-4

[2] G. Esteves, F. Fidalgo, N. Cruz, and J. Simão, “Long-Range Wide Area Network Intrusion Detection at the Edge,” *IoT*, Vol. 5, No. 4, Art. No. 4, Dec. 2024, doi: 10.3390/iot5040040

[3] S. Milani, I. Chatzigiannakis, D. Garlisi, M. D. Fraia, and P. Pisani, “Enabling Edge processing on LoRaWAN architecture,” in *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, Oct. 2023, pp. 1-3, doi: 10.1145/3570361.3614074

[4] P. Spadaccino and F. Cuomo, “Intrusion Detection Systems for IoT: opportunities and challenges offered by Edge Computing and Machine Learning,” Apr. 14, 2022, *arXiv*: arXiv:2012.01174, doi: 10.48550/arXiv.2012.01174

[5] A. Proto, C. Miers, and T. Carvalho, "An Intrusion Detection Architecture Based on the Energy Consumption of Sensors Against Energy Depletion Attacks in LoRaWAN:," in *Proceedings of the 9th International Conference on Internet of Things, Big Data and Security*, Angers, France: SCITEPRESS - Science and Technology Publications, 2024, pp. 268-275, doi: 10.5220/0012703400003705

[6] M. Chen, L. Mokdad, J. B. Othman, and J.-M. Fourneau, "LoRaLOFT-A Local Outlier Factor-based Malicious Nodes detection Method on MAC Layer for LoRaWAN," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, 2022, pp. 2026-2031, doi: 10.1109/GLOBECOM48099.2022.10000852

[7] M. Babazadeh, "LoRa-Based Anomaly Detection Platform: Center and Sensor-Side," *IEEE Sens. J.*, Vol. 20, No. 12, pp. 6677-6684, June 2020, doi: 10.1109/JSEN.2020.2976650

[8] A. Kurniawan and M. Kyas, "Machine Learning Models for LoRa Wan IoT Anomaly Detection," in *2022 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, 2022, pp. 193-198 doi: 10.1109/ICACSIS56558.2022.9923439

[9] "LoRaWAN® Specification v1.0.3," LoRa Alliance®. Accessed: May 20, 2025 [Online] Available: https://hz1.37b.myftpupload.com/resource_hub/lorawan-specification-v1-0-3/

[10] "LoRaWAN® Specification v1.1." Accessed: May 20, 2025 [Online] Available: https://resources.lora-alliance.org/technical-specifications/lorawan-specification-v1-1

[11] "ChirpStack open-source LoRaWAN Network Server." Accessed: May 26, 2025 [Online] Available: https://www.chirpstack.io/

[12] A. Povalac, J. Kral, H. Arthaber, O. Kolar, and M. Novak, "Exploring LoRaWAN Traffic: In-Depth Analysis of IoT Network Communications," *Sensors*, Vol. 23, No. 17, Art. No. 17, Jan. 2023, doi: 10.3390/s23177333

[13] *Lora-net/picoGW_hal*. (Aug. 22, 2024) C. LoRa®. Accessed: June 02, 2025. [Online]. Available: https://github.com/Lora-net/picoGW_hal

[14] "Scapy." Accessed: June 02, 2025 [Online] Available: https://scapy.net/

[15] "Elastic Stack: (ELK) Elasticsearch, Kibana & Logstash," Elastic. Accessed: May 26, 2025 [Online] Available: https://www.elastic.co/elastic-stack

[16] "Protocol Buffers." Accessed: May 26, 2025 [Online] Available: https://protobuf.dev/

[17] "Loracrack/loracrack_decrypt.c at master applied-risk/Loracrack," GitHub. Accessed: May 26, 2025 [Online] Available: https://github.com/applied-risk/Loracrack/blob/master/loracrack_decrypt.c