Towards a Protocol-Aware Intrusion Detection System for LoRaWAN Networks

Zsolt Bringye , Rita Fleiner * , Eszter Kail

Posted Date: 5 February 2026

doi: 10.20944/preprints202602.0072.v1

Keywords: LoRaWan security; anomaly detection; SIEM; OTAA; IDS; digital twin; FSM

 doi:10.20944/preprints202602.0072.v1

Article

Towards a Protocol-Aware Intrusion Detection System for LoRaWAN Networks

Zsolt Bringye [1], Rita Fleiner [1,*] and Eszter Kail [1,2]

[1] Obuda University, Neumann Faculty of Informatics
[2] HUN-REN Institute for Computer Science and Control (HUN-REN SZTAKI), Hungarian Research Network
* Correspondence: fleiner.rita@nik.uni-obuda.hu

Abstract

The increasing reliance of Internet of Things (IoT) applications on low-power wide-area network technologies, particularly LoRaWAN, has amplified the need for intrusion detection approaches that go beyond attack-specific signatures and generic traffic anomalies. Existing IoT intrusion detection systems are often tailored to individual threat scenarios or rely on statistical indicators, which limits their ability to capture protocol-level misuse in a systematic and interpretable manner. This paper addresses this gap by proposing a methodology for protocol-aware anomaly detection based on a digital twin abstraction of LoRaWAN communication behavior. The approach models the Over-The-Air Activation (OTAA) procedure as a finite-state machine that serves as a lightweight, protocol-specific digital twin, encoding expected message sequences and specification-driven constraints. Rather than targeting individual attacks, observed network events are continuously validated against the modeled state evolution, enabling the identification of deviations that indicate anomalous or non-conformant behavior. Illustrative examples include replay attempts, integrity violations, and inconsistencies in protocol parameters, although the framework is not limited to predefined attack categories. The results demonstrate that state-machine-based digital twins provide a structured and extensible foundation for intrusion detection and can be integrated into SOC (Security Operation Center) oriented monitoring environments. Overall, the study highlights the methodological advantages of digital-twin-driven, state-aware detection for improving protocol compliance monitoring and interpretability in LoRaWAN-based IoT networks. Unlike prior LoRaWAN IDS approaches, the proposed model enables the detection of protocol-conformant yet semantically invalid behaviors that remain invisible to packet-centric or statistical detectors.

Keywords: LoRaWan security; anomaly detection; SIEM; OTAA; IDS; digital twin; FSM

1. Introduction

The increasing deployment of LoRaWAN-based Internet of Things (IoT) systems in industrial, critical infrastructure, and smart-city environments introduces security challenges that are not adequately addressed by conventional intrusion detection approaches.

While LoRaWAN provides cryptographic protection and energy-efficient communication, its protocol design, constrained observability, and distributed architecture complicate the detection of protocol misuse and stealthy attacks, particularly during security-critical phases such as device activation.

In contrast to traditional IT networks, LoRaWAN traffic is sparse, largely encrypted, and highly state-dependent. As a result, many existing IoT intrusion detection systems rely on lightweight statistical indicators or attack-specific heuristics derived from limited metadata. Although such approaches are effective for identifying volumetric anomalies or radio-level disturbances, they are insufficient for capturing protocol-level violations that remain syntactically valid but semantically inconsistent with the expected protocol behavior.

 doi:10.20944/preprints202602.0072.v1

Beyond data-driven and heuristic intrusion detection approaches, several studies have demonstrated the effectiveness of model-driven and specification-based techniques for security monitoring in complex cyber-physical and industrial systems. In such settings, formal behavioral models enable systematic reasoning about correctness and the detection of deviations from expected operation without relying on learning-based methods [1,2]. These results suggest that explicit protocol and state modeling can provide strong foundations for interpretable and verifiable intrusion detection in distributed, resource-constrained environments.

In our earlier work, we initiated the construction of a SOC (Security Operation center) compatible monitoring pipeline for LoRaWAN networks, demonstrating how telemetry from selected points of the communication chain can be ingested, enriched, and correlated within a centralized SIEM environment [3]. That study identified multiple observation points across the LoRaWAN pipeline and implemented a practical preprocessing layer capable of detecting metadata inconsistencies and supporting payload decoding when cryptographic context was available. While the monitoring pipeline was not yet complete in terms of full multi-point coverage, the results confirmed that protocol-aware analysis of LoRaWAN traffic is feasible within standard SOC workflows. However, the previous work also revealed a fundamental limitation: without an explicit model of the LoRaWAN protocol state, many security-relevant deviations cannot be systematically identified.

Attacks and misconfigurations during the Over-The-Air Activation (OTAA) procedure, such as replayed join messages, nonce reuse, or abnormal retry patterns, often manifest as violations of protocol progression rather than malformed packets or isolated anomalies. Detecting such behavior requires reasoning about when a message is valid, not only what it contains. Motivated by these observations, this paper introduces a protocol-aware intrusion detection methodology based on a finite-state digital twin of the LoRaWAN OTAA procedure. The proposed model formalizes the expected sequence of join-related messages, timing constraints, and state transitions, enabling observed events to be continuously validated against the protocol specification. Rather than targeting predefined attack signatures, the approach detects deviations from legitimate protocol evolution, providing an interpretable and extensible foundation for anomaly detection.

By building on the previously established SOC-integrated monitoring pipeline, the state-machine-based digital twin presented in this work complements existing observability mechanisms with formal protocol semantics. Together, these components enable systematic detection of OTAA-level misuse and lay the groundwork for extending state-aware intrusion detection to other phases of the LoRaWAN protocol.

The main contributions of this paper are as follows:

- **Protocol-Aware Formalization of the OTAA Procedure.** We present a finite state machine (FSM)–based formalization of the LoRaWAN Over-The-Air Activation (OTAA) procedure, explicitly modeling valid protocol states, message sequences, and timing constraints. Unlike packet-centric or statistical approaches, the proposed model captures the semantic correctness of protocol progression.
- **State-Based Characterization of OTAA Security Weaknesses.** We systematically reinterpret known OTAA weaknesses, such as DevNonce reuse, replay conditions, join flooding, and state desynchronization, such as violations of expected state transitions rather than isolated packet anomalies. This perspective enables structured and interpretable detection of protocol misuse.
- **Integration with SOC-Oriented LoRaWAN Monitoring Pipelines.** Building on our earlier work on SOC-compatible LoRaWAN telemetry ingestion, we position the FSM model as a logical validation layer operating on enriched protocol metadata. The proposed approach complements existing SIEM-based observability mechanisms by adding formal protocol semantics without assuming full payload visibility.
- **Digital-Twin–Style Detection Model for Encrypted LoRaWAN Traffic.** We demonstrate how a state-machine-based digital twin of the OTAA procedure enables intrusion detection even under

 doi:10.20944/preprints202602.0072.v1

constrained observability and encrypted communication, relying on protocol-level metadata and state consistency rather than deep packet inspection.

- **Foundations for Extensible State-Aware IDS Design.** While this work focuses on OTAA, the presented methodology is intentionally generic and can be extended to other phases of the LoRaWAN protocol or to similar low-power IoT systems where security-relevant behavior is inherently state-dependent.

The remainder of this paper is organized as follows. Section 2 provides background on LoRaWAN communication, with particular emphasis on the Over-The-Air Activation (OTAA) procedure and its protocol-level observability. Section 3 reviews related work, covering known security weaknesses of OTAA as well as existing intrusion detection and digital-twin-based approaches in IoT networks, and positions the present work within this landscape. Section 4 introduces the proposed methodology, including the threat model, observability assumptions, and the protocol-aware Digital Twin realized as a finite-state machine. This section details the hierarchical rule system, formal semantics, and state transition logic used for protocol-conformance validation. Section 5 describes the experimental setup and system implementation, including the LoRaWAN testbed, monitoring points, preprocessing pipeline, and SIEM integration. Section 6 presents the evaluation results obtained from controlled OTAA scenarios, illustrating FSM state evolution, detection outcomes, and SOC-level interpretability. Finally, Section 7 discusses the implications, limitations, and future extensions of the proposed approach, and Section 8 concludes the paper.

2. Background

2.1. LoRa, LoRaWAN

LoRaWAN [4] (Long Range Wide Area Network) is a low-power wide-area networking protocol designed to support long-range, energy-efficient communication for large-scale Internet of Things (IoT) deployments. It operates on top of LoRa radio technology [5], which employs chirp spread spectrum modulation to achieve robust communication under low signal-to-noise conditions while enabling multi-year battery lifetimes for resource-constrained end devices.

Beyond the physical layer, LoRaWAN defines both the communication protocol and the system architecture governing device-to-cloud interaction. The network follows a star-of-stars topology in which end devices transmit LoRa frames directly to one or more gateways. Gateways act as transparent forwarders, relaying received radio packets to backend servers over IP-based networks without maintaining LoRaWAN protocol state. This design enables redundancy, multi-gateway reception, and scalable network operation.

The backend infrastructure is logically decomposed into functional components with clearly separated responsibilities. The Network Server performs MAC-layer processing, including frame validation, message deduplication, adaptive data rate control, downlink scheduling, and device state tracking. Application Servers process application-layer payloads and expose data to end-user services. Newer LoRaWAN specifications [6] formalize the Join Server as a dedicated component responsible for authentication and join-related key management, reinforcing security separation within the architecture.

LoRaWAN defines multiple operational modes for end devices to balance energy efficiency and downlink availability. **Class A** operation, which is mandatory for all devices, provides the lowest energy consumption by opening short receive windows only after uplink transmissions. **Class B** and **Class C** introduce progressively more predictable downlink reception at the cost of increased energy usage. These class-dependent behaviors result in distinct communication patterns and timing characteristics that are observable at the network level.

Communication in LoRaWAN relies on a small set of well-defined message types, including join messages and data frames. Data frames follow a layered structure consisting of a MAC header, a MAC payload, and a Message Integrity Code (MIC). The MAC payload contains addressing and control information, frame counters, optional MAC commands, and the encrypted application payload.

 doi:10.20944/preprints202602.0072.v1

This structure enables both application data transfer and protocol control signaling within a single cryptographically protected frame. At the physical layer, the LoRaWAN payload is encapsulated within a LoRa PHY frame, whose size and format depend on regional parameters and data rate settings. Security is a core design principle of LoRaWAN. The aforementioned message structure is depicted in Figure 1.

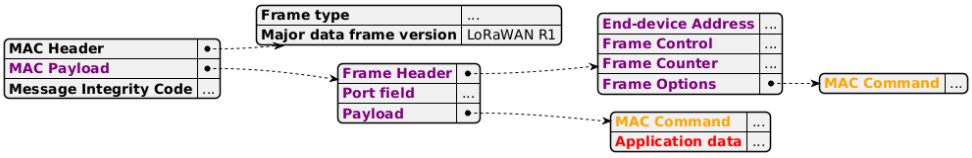

Figure 1. Structure of Data Frame [3].

The protocol employs symmetric cryptography and session-based keys to ensure confidentiality and integrity of communication. Message integrity is provided by the MIC appended to each frame, allowing detection of unauthorized modifications. The exact fields included in the MIC computation depend on the message type, implicitly binding integrity verification to the protocol state.

Device activation is performed either through Over-the-Air Activation (OTAA) or Activation by Personalization (ABP). OTAA dynamically establishes a session and derives fresh session keys, while ABP relies on statically provisioned parameters and does not involve a runtime join exchange. Because join procedures and key derivation differ across specification versions, and to remain consistent with both our experimental environment and prevailing deployments, this work adopts the LoRaWAN 1.0.3 specification. Although LoRaWAN 1.1 introduces notable security improvements, version 1.0.3 remains widely deployed due to its maturity, compatibility with legacy devices, and continued support in commonly used network server implementations such as ChirpStack. Basing the analysis on LoRaWAN 1.0.3 therefore allows the presented results to reflect realistic operational conditions.

In addition to the transmitted payload, LoRaWAN communication is enriched with contextual metadata during reception and forwarding. Gateways typically attach radio-level attributes such as received signal strength (RSSI), signal-to-noise ratio (SNR), frequency, channel, and timestamps to each received frame, and periodically report their own operational status. Network servers further augment messages with device registry information and session state. When join context and cryptographic keys are available, application payloads can be decrypted, enabling deeper inspection. This multi-layer enrichment across radio, MAC, and application layers provides a rich observation space and forms the foundation for the state-aware modeling and anomaly detection approach presented in the remainder of this paper.

2.2. OTAA

Over-the-Air Activation (OTAA) is the preferred activation mechanism in LoRaWAN, enabling dynamic session establishment and per-session key derivation. The join procedure defines the initial protocol state of an end device and establishes the cryptographic context for subsequent communication, making it a security-critical phase of the protocol.

The OTAA process is initiated by the end device through a `Join Request` uplink message. This message contains the device and application identifiers (DevEUI, JoinEUI/AppEUI) and a device-generated DevNonce, which is intended to be unique for each join attempt. The `Join Request` payload is not encrypted but is protected by a Message Integrity Code (MIC) computed using the root application key (AppKey), ensuring authenticity and integrity. The presence, reuse, or timing of DevNonce values is therefore a key observable during activation. Upon successful validation, the network responds with a `Join Accept` downlink message, which is encrypted and authenticated using the AppKey. The `Join Accept` assigns a dynamic DevAddr and provides essential network parameters, including AppNonce, NetID, DLSettings, and RXDelay, with an optional CFList depending on regional configuration. Using the exchanged nonces and the AppKey, the device derives session keys for MAC-layer integrity and application payload encryption.

 doi:10.20944/preprints202602.0072.v1

In LoRaWAN version 1.0.3 rejoining is typically realized by repeating the standard OTAA procedure rather than using explicit rejoin message types. Rejoin events may occur after device reboot, session expiration, or prolonged communication failures. While such behavior can be benign, repeated or irregular join attempts, inconsistent timing, or nonce reuse may also indicate misconfiguration or malicious interference.

From a detection perspective, the OTAA procedure is inherently state-dependent. The legitimacy of a `Join Request` or `Join Accept` message cannot be assessed in isolation, but only in relation to prior messages, timing constraints, and the expected protocol phase of the device. This makes OTAA particularly suitable for state-aware analysis, as many security-relevant deviations manifest as violations of expected protocol progression rather than malformed packets. In the following section, we therefore examine known weaknesses and attack vectors affecting the OTAA process, which motivate the finite state machine (FSM) model introduced later in this paper.

3. Related Work

3.1. Security Weaknesses and Attack Surfaces of OTAA

The Over-the-Air Activation (OTAA) procedure plays a central role in establishing secure LoRaWAN communication, however, while intended to offer superior security over Activation by Personalization (ABP), the design of LoRaWAN version 1.0.x introduces several security-relevant weaknesses and vulnerabilities [7], [8]. These weaknesses primarily stem from the limited entropy of nonces, the lack of cryptographic binding between join messages, and the centralized trust model where the network server manages all encryption keys [9]. Many of these issues do not arise from malformed packets or broken cryptography, but from violations of expected protocol sequencing, timing, or value reuse. As a result, they are particularly difficult to detect using stateless or signature-based approaches.

DevNonce Reuse and Replay Conditions

In LoRaWAN 1.0.3, the DevNonce used in `Join Requests` messages is a 16-bit value generated by the end device and intended to be unique per join attempt [10]. This design is inherently susceptible to the birthday paradox, where the probability of generating a duplicate nonce increases rapidly over the device's lifetime, estimated at a 10-year operational span [9]. In practice, constrained devices may reuse DevNonce values due to reboots, counter resets, or poor randomness. While the specification mandates DevNonce uniqueness, enforcement relies on network-side tracking of previously observed values. Furthermore, many low-cost IoT devices, such as those utilizing the SX1272 transceiver, rely on the least significant bits (LSB) of RSSI (Received Signal Strength Indicator) measurements as an entropy source for random number generation [11]. Experimental data indicate that such hardware-based generators often lack health tests and exhibit predictable patterns, which can be further compromised through targeted RF jamming [9]. DevNonce reuse enables replay-based attack scenarios, where previously captured `Join Requests` are retransmitted to trigger unintended join behavior or desynchronize session state. Importantly, such messages are syntactically valid and protected by a correct MIC, making them indistinguishable from legitimate traffic when evaluated in isolation. Detection therefore requires stateful tracking of DevNonce values across join attempts and correlation with prior protocol history.

Join Requests Flooding and Activation Abuse

The OTAA procedure is uplink-driven and may be repeatedly triggered by an end device in the absence of a successful `Join Accept`. This behavior is expected under benign conditions such as downlink loss or temporary network unavailability. However, excessive or patterned `Join Requests` transmissions can also be exploited to cause resource exhaustion at the network server, increase gateway load, or disrupt legitimate device activation.

 doi:10.20944/preprints202602.0072.v1

Also, in LoRaWAN version 1.0.x, the `Join Requests` message is transmitted in plaintext, which exposes the device's identifiers (DevEUI and AppEUI/JoinEUI) to passive observers. This lack of confidentiality, combined with the protocol's retry mechanisms, enables several abuse scenarios. An adversary can perform a self-replay attack by selectively jamming the Network Server's (NS) `Join Accept` response. Following the specification, the end device will retransmit the original `Join Requests` using the same DevNonce, allowing the attacker to force the device into repeating this cycle until its daily message quota (e.g., 14 packets per day) is depleted, effectively rendering the device unreachable [11]. Such abuse does not necessarily violate packet-level constraints. Instead, it manifests as abnormal activation frequency, inconsistent timing, or prolonged residence in the joining state. Distinguishing benign retries from malicious behavior requires reasoning about expected retry behavior, timing constraints, and device state evolution.

Join Accept Manipulation and Desynchronization Effects

A fundamental flaw in LoRaWAN version 1.0.x is that the `Join Accept` message is not cryptographically bound to the preceding Join Request. The payload and Message Integrity Code (MIC) of the `Join Accept` are independent of the DevNonce provided by the device. This vulnerability allows an adversary to perform a `Join Accept` replay attack, where a previously captured valid `Join Accept` is retransmitted in response to a new `Join Request`.

This vulnerability allows an adversary to perform a `Join Accept` replay attack, where a previously captured valid `Join Accept` is retransmitted in response to a new `Join Request` [9]. Loss or manipulation of `Join Accept` messages, whether due to radio conditions or adversarial interference, can lead to state desynchronization [9], where the network considers a device joined while the device remains unactivated. Such desynchronization may cause repeated join attempts, inconsistent session key usage, or unexpected message sequences. From a monitoring perspective, these situations appear as valid messages occurring in unexpected protocol states, rather than explicit errors.

Timing and State Transition Violations

The OTAA procedure imposes implicit timing relationships between uplink `Join Requests` and downlink `Join Accept` reception windows. Deviations from expected timing, such as texttt{Join Accept messages observed outside valid receive windows or join related messages appearing after an established session, represent violations of protocol progression rather than structural anomalies. Formal verification using Timed Automata has shown that even minor timing drifts can lead to the loss of downlink synchronization [8]. These timing-based inconsistencies are particularly relevant in environments with multiple gateways and asynchronous forwarding, where reordered or duplicated observations may occur. Correct interpretation therefore requires explicit modeling of allowed state transitions and their temporal constraints.

Across these scenarios, a common pattern emerges: OTAA-related attacks and misconfigurations rarely produce invalid packets. Instead, they exploit ambiguities in protocol state, value reuse, and timing behavior. As a result, purely stateless detection mechanisms or metadata thresholding are insufficient to capture the full attack surface of the OTAA procedure. These observations motivate a detection strategy that explicitly models OTAA as a sequence of well-defined states and transitions, with associated constraints on message content and timing. In the following section, we formalize the OTAA procedure using a finite state machine (FSM) and demonstrate how state-aware evaluation enables systematic identification of the vulnerabilities discussed above.

3.2. Intrusion Detection Systems and Digital Twin Solutions in IoT Networks

Intrusion Detection Systems (IDS) in Internet of Things (IoT) environments have evolved into a broad spectrum of architectural and methodological approaches, reflecting the diversity and complexity of the systems they aim to protect. Depending on latency, privacy, and computational constraints, IDS solutions can be deployed across edge, fog, and cloud layers [12,13].

 doi:10.20944/preprints202602.0072.v1

Beyond architectural placement, IDS solutions are commonly classified according to their detection logic. In addition to signature- and anomaly-based techniques, stateful protocol analysis (often referred to as specification-based intrusion detection) relies on formal protocol specifications and explicit state tracking to identify deviations from expected behavior [14]. This class of IDS is particularly well suited to protocol-driven and wireless communication systems, where security-relevant anomalies often manifest as violations of message ordering, timing constraints, or state-dependent invariants rather than isolated packet-level features.

However, the growing protocol diversity and stateful nature of IoT communications pose significant challenges for conventional IDS designs, particularly in low-power and delay-sensitive networks, where scalability and protocol specificity are critical.

To address these limitations, several studies have explored structured, protocol-aware detection mechanisms that incorporate formal models such as Finite State Machines (FSMs) or Extended FSMs (EFSMs). Beyond purely statistical detection, explicit state modeling enables IDS solutions to reason about protocol semantics rather than aggregate traffic behavior. By encoding valid message sequences, timing constraints, and state-dependent invariants, FSM- and EFSM-based approaches can detect subtle violations that remain indistinguishable at the packet or feature level. This capability is particularly important in protocols such as LoRaWAN, where many security-relevant deviations manifest only across multiple messages or reception windows.

In parallel, Digital Twin (DT) paradigms have gained attention as a means to maintain a virtual representation of device behavior or protocol execution, enabling the detection of deviations between expected and observed states. While the notion of a digital twin varies across application domains, a common characteristic is the continuous synchronization between the physical system and its virtual counterpart. In many IoT security applications, digital twins primarily serve as contextual mirrors or simulation environments. In contrast, protocol-aware digital twins emphasize real-time synchronization and enforcement, allowing deviations from expected protocol behavior to be detected as part of normal system operation. These approaches are especially relevant for lightweight IoT technologies such as LoRaWAN, where payload inspection is limited, but protocol-level inconsistencies, such as unauthorized joins, replay attempts, or timing violations, which can be effectively identified through state-driven logic.

State-Based IDS Approaches for LoRaWAN

The stateful nature of the LoRaWAN Over-The-Air Activation (OTAA) procedure has motivated several intrusion detection approaches based on protocol modeling. Danish et al. [15] propose LIDS, an IDS that monitors the OTAA join process to detect jamming attacks. Their method analyzes DevNonce values using statistical divergence metrics, such as Kullback–Leibler divergence and Hamming distance, achieving high detection accuracy. While their approach implicitly captures protocol state through statistical modeling, it does not explicitly represent the join procedure as a formal state machine, which limits interpretability and protocol coverage.

Similarly, Horák et al. [16] investigate denial-of-service attacks targeting the OTAA procedure by analyzing `Join-Request` patterns at the gateway level. Their method relies on DevNonce randomness analysis and static statistical indicators derived from gateway logs. Although state-aware in spirit, their solution does not model the full protocol state evolution and remains limited to partial observability at a single monitoring point. These studies illustrate that while statistical methods can detect certain anomalies, they struggle to distinguish between benign retransmissions, gateway artifacts, and protocol-level violations without explicit state modeling.

Beyond LoRaWAN-specific solutions, several studies demonstrate the effectiveness of specification-based and automata-driven IDS in constrained IoT environments. Fu et al. [17] introduce an automata-based detection framework for heterogeneous IoT protocols, using labeled transition systems to identify replay, jamming, and message forgery attacks. Likewise, Le et al. [18] propose an EFSM-based IDS for RPL (Routing Protocol for Low Power Lossy Network) networks, where deviations from expected protocol state transitions are used to detect routing-level attacks. Although these works

 doi:10.20944/preprints202602.0072.v1

target protocols other than LoRaWAN, they underline the value of explicit protocol state modeling for detecting logic-level attacks in low-power and lossy networks.

Digital Twin–Based Anomaly Detection in IoT

Fuller et al. [19] provide a comprehensive conceptual overview of digital twin technologies, explicitly distinguishing between digital models, digital shadows, and fully integrated digital twins based on the directionality and continuity of data exchange between the physical and virtual systems. This distinction is particularly relevant for protocol-centric IoT systems, where the digital twin often acts as a behavioral and state-based reference model rather than a physics-based replica. In such settings, the primary value of a digital twin lies in its ability to maintain synchronized protocol state and expected behavior, which can be leveraged for monitoring and anomaly detection.

Eckhart and Ekelhart in [20]propose a security-aware digital twin framework for cyber-physical systems, in which the virtual replica is automatically derived from formal system specifications and continuously synchronized with the physical system. In their approach, the digital twin serves as a reference model for monitoring and intrusion detection, enabling the identification of deviations from expected behavior without relying on learning-based techniques. This work demonstrates that specification-driven, state-aware detection mechanisms can provide effective protection, however, the proposed framework remains conceptual and is not integrated into an operational intrusion detection system or SOC-oriented detection pipeline.

Homaei et al. [21] present a multi-layered security architecture for smart water infrastructures, combining a LoRaWAN-based sensor network with machine learning driven IDS components and a digital twin. In their system, the digital twin primarily serves as a trusted representation of infrastructure behavior and a decision-support tool, while intrusion detection is performed by statistical and learning-based models. As such, the digital twin remains auxiliary to detection rather than constituting the detection logic itself.

El-Hajj et al. [22] integrate a real-time digital twin with a Snort-based IDS in smart city environments, using the twin to mirror device-level metrics such as CPU usage and connectivity state. While effective against volumetric network attacks, the digital twin in their framework does not encode protocol semantics and remains decoupled from intrusion detection decisions.

Schösser et al. [23] employ a digital twin of the radio environment to detect jamming attacks by comparing predicted and observed signal strength values. Their approach demonstrates strong performance at the physical layer but relies on a static twin model and does not incorporate network or protocol-level state.

Positioning of the Present Work

Building on digital twin taxonomies that distinguish between static digital models, digital shadows, and fully synchronized digital twins, this work adopts a protocol-level interpretation of the digital twin concept. In the context of protocol-centric IoT systems such as LoRaWAN, the digital twin is intended to represent expected behavioral and state evolution of the communication protocol.

In contrast to existing approaches that employ digital twins primarily as passive monitoring abstractions, simulation environments, or auxiliary analysis tools, we integrate the digital twin directly into the intrusion detection process. The proposed digital twin is realized as an executable Finite State Machine (EFSM) that formalizes the LoRaWAN OTAA join procedure and continuously validates real-world communication against explicitly specified protocol state transitions.

By embedding protocol semantics, timing constraints, and security invariants into the digital twin itself, the model serves as the core detection engine rather than a supporting component. The integration of multi-layer telemetry sources, including radio, gateway, and network server observations enables real-time detection of replay attacks, timing violations, and protocol misuse. As a result, the proposed approach bridges formal protocol modeling and operational intrusion detection, providing a lightweight, interpretable, and SOC-compatible IDS framework tailored to LoRaWAN deployments.

 doi:10.20944/preprints202602.0072.v1

4. Methodology and Prior Work

4.1. Threat Model and Observability Assumptions

This work considers realistic threat scenarios in operational LoRaWAN deployments, where end devices operate unattended and communicate over a shared wireless medium. The attacker is assumed to have access to the wireless channel and limited network-level interaction, enabling passive traffic observation and basic active interference such as message replay, timing manipulation, or protocol misuse. Direct access to backend infrastructure, gateway internals, or cryptographic root keys is explicitly out of scope.

A fundamental challenge in LoRaWAN security analysis is constrained observability in adversarial settings. Due to end-to-end encryption and integrity protection, application payloads and Message Integrity Codes (MICs) cannot be evaluated by an external observer without cryptographic context. Consequently, intrusion detection mechanisms that operate without access to session keys must primarily rely on protocol-level metadata and observable behavior, including message types, timing relationships, frame counters, radio-layer attributes (RSSI, SNR (Signal-to-Noise Ratio), frequency), and gateway reception context.

At the same time, the proposed Digital Twin–based IDS is not inherently limited to metadata only analysis. When cryptographic session context is available, such as in network-server–side deployments, controlled testbeds, or authorized security monitoring environments, the Digital Twin can fully decode, authenticate, and formally analyze all protocol fields, including encrypted payloads and integrity checks. This dual-mode capability allows the framework to operate under realistic observability constraints while enabling deeper, specification-level analysis whenever cryptographic material is accessible.

In practice, this corresponds to typical defender-side deployments, where the IDS operates alongside or within trusted backend components such as the network server or SOC infrastructure.

Crucially, the security relevance of both metadata and decoded protocol fields is inherently state-dependent: identical values or fields may indicate benign or malicious behavior depending on the current phase of the protocol. This observation motivates state-aware detection mechanisms that explicitly reason about protocol progression rather than isolated packets.

4.2. Prior Work: SOC-Compatible Telemetry Pipeline

In our earlier work, we addressed the problem of observability by designing a SOC compatible telemetry pipeline for LoRaWAN networks, without enforcing explicit protocol semantics [3].

While this prior framework demonstrated that security-relevant inconsistencies can be surfaced from protocol metadata alone, detection remained largely correlation-based. In particular, protocol violations spanning multiple messages or timing windows could not be robustly identified without an explicit representation of protocol state.

4.3. Approach Overview: Protocol-Aware Digital Twin as Detection Core

Building on the established telemetry pipeline, the present work introduces a protocol-aware Digital Twin as the core detection mechanism. The Digital Twin maintains an internal communication context analogous to that of the LoRaWAN network server, tracking protocol-relevant state information and cryptographic parameters across events.

Unlike passive monitoring approaches, the Digital Twin actively enforces protocol semantics. Incoming events are evaluated against the expected protocol progression, allowing the system to detect violations such as invalid message ordering, nonce reuse, or timing inconsistencies that cannot be reliably identified through stateless correlation alone.

This design shifts intrusion detection from packet-level anomaly spotting to protocol-level conformance checking, providing both increased detection precision and improved interpretability.

 doi:10.20944/preprints202602.0072.v1

4.4. Finite State Machine–Based IDS

To strengthen the security and correctness of the LoRaWAN Over-The-Air Activation (OTAA) procedure, we model protocol execution using a rule-based finite state machine (FSM) that captures both valid protocol progression and security-relevant failure modes. Beyond the nominal join workflow, the model explicitly encodes constraints related to malformed messages, protocol misuse, replay attempts, and timing violations commonly exploited in real-world LoRaWAN attacks.

The FSM is implemented using a domain-specific language (DSL) embedded in Python [24], allowing protocol behavior to be expressed declaratively as guarded state transitions. Each incoming event, either a message reception or a timer expiration, is evaluated against a structured set of protocol invariants covering radio-layer compliance, state-dependent message ordering, cryptographic integrity, replay protection, and reception window timing. Events violating these constraints are explicitly rejected with a well-defined outcome, while generating structured telemetry suitable for integration into SIEM-based monitoring workflows.

From an architectural perspective, the FSM realizes a protocol-level Digital Twin that actively enforces LoRaWAN semantics rather than passively observing traffic. Validation rules are applied in a layered hierarchy (Level 0 - physical and radio-layer compliance, level 1 - protocol state guarding, level 2 - cryptographic and replay security checks, and level 3 -functional flow execution) ensuring that higher-level logic is evaluated only after lower-level constraints are satisfied. The FSM is defined independently of any specific implementation, allowing the same formal model to support offline analysis and verification, as well as runtime enforcement in monitoring and testing environments.

4.5. Formal Description of the Digital Twin Rule Engine

The formal rule system defined below serves two complementary purposes. First, it provides a precise, unambiguous specification of valid and invalid protocol behavior during the OTAA procedure. Second, it constitutes the executable logic of the Digital Twin based IDS, enabling runtime evaluation of incoming events against protocol semantics.

The rules are organized into hierarchical validation levels, reflecting the layered structure of the LoRaWAN protocol. Each level captures a distinct class of constraints, allowing violations to be localized to specific protocol layers and improving the interpretability of detected anomalies.

4.5.1. Notation and Semantics

To enable precise reasoning and unambiguous interpretation, the Digital Twin based IDS is formalized as an event driven finite state machine. In the following, we introduce the notation and formal semantics used to describe events, state evolution, and rule evaluation.

Formally, the behavior of the Digital Twin is defined as an event driven state machine, where events are denoted with $e \in \mathcal{E}$. The event set is partitioned into message-related and timer-related events as

$$\mathcal{E} = \mathcal{E}_{\text{msg}} \cup \mathcal{E}_{\text{tmr}}. \tag{1}$$

A message event $e \in \mathcal{E}_{\text{msg}}$ is associated with an incoming frame m, which is characterized by attributes such as message (type), operating frequency (freq), data rate (dr), and cryptographic metadata. Timer events $e \in \mathcal{E}_{\text{tmr}}$ represent the expiration of previously scheduled protocol timers.

The system state is represented by the state vector

$$v \triangleq \langle state, history \rangle, \tag{2}$$

where *state* denotes the current protocol state and *history* captures the relevant communication context accumulated from prior events.

The communication context maintained by the Digital Twin captures all cryptographic and stateful parameters required by the LoRaWAN protocol. In particular, it includes the DevNonce generated by the end device and the AppNonce provided in the join-accept message, which jointly ensure freshness

 doi:10.20944/preprints202602.0072.v1

and replay protection of the join procedure. The context further stores the Application Key (AppKey), which acts as master cryptographic key in LoRaWAN and is used as input to the key derivation functions defined by the specification. From the AppKey, the Network Session Key (NwkSKey) and the Application Session Key (AppSKey) are derived upon successful completion of the join procedure and retained as part of the communication context for subsequent message integrity verification and payload encryption.

This communication context is actively maintained and updated by the Digital Twin and directly influences rule activation and admissible state transitions, rather than serving as a passive record of observed traffic.

State transitions are expressed as guarded rules of the form

$$(v, e) \xrightarrow{\text{rule}} v', \tag{3}$$

where e is the triggering event and v' denotes the updated state vector resulting from the application of the corresponding rule. A rule is enabled if its guard condition evaluates to true for the current state and event, in which case the corresponding state update is applied.

The rule engine processes incoming events using a hierarchical evaluation strategy. By structuring validation rules into successive levels, the Digital Twin mirrors the layered nature of the LoRaWAN protocol while enabling early rejection of non-compliant traffic. Low-level rules capture fundamental radio and timing constraints, which are inexpensive to evaluate and allow malformed or physically implausible frames to be filtered before more complex analysis is performed. Higher-level rules operate on increasingly stateful and contextual information, incorporating security-oriented guardrails and protocol flow validation that require knowledge of prior events and system history. This design not only reduces unnecessary processing overhead but also improves the interpretability of detected anomalies by localizing violations to specific protocol layers. As a result, the proposed digital twin based IDS supports protocol-aware, explainable intrusion detection, while maintaining extensibility and alignment with LoRaWAN 1.0.3 specification.

In the following, we present the formal specification of the proposed rule system, structured according to its hierarchical evaluation levels.

4.5.2. States, Events, and Timers

In our Digital Twin state ranges over a finite set of protocol phases defined by the OTAA procedure. The explicitly modeled states and their definitions are listed in Table 1

Table 1. Finite State Machine states of the LoRaWAN OTAA Digital Twin.

State	Description
NDEF	Idle or reset state, no active join procedure
JOINING_RX1DELAY	Waiting for the opening of the RX1 reception window
JOINING_RX1	RX1 reception window active
JOINING_RX2DELAY	Waiting for the opening of the RX2 reception window
JOINING_RX2	RX2 reception window active
JOINED	Device successfully activated and joined to the network

For compactness, we additionally employ state sets that represent semantically equivalent acceptance conditions listed in Table 2.

Table 2. Composite FSM states used for protocol guard conditions.

Composite State	Description
CAN_REQUEST_JOIN	Set of states in which a `Join Request` message is admissible
CAN_ACCEPT_JOIN	Set of states in which a `Join Accept` message may be processed

 doi:10.20944/preprints202602.0072.v1

Message events correspond to the reception of LoRaWAN frames and carry a message type $m(e).\text{type} \in \{\text{JR} - \texttt{Join request}, \text{JA} - \texttt{Join Accept}, \text{DATA}\}$.

Timer events model protocol-defined timing boundaries and reception window timeouts. The OTAA model uses the timer identifiers as: TOUT_RX1START, TOUT_RX1END, TOUT_RX2START, TOUT_RX2END, which correspond to the opening and closing instants of the RX1 and RX2 receive windows. Timer expiration is denoted by $\text{Expired}(\cdot)$, while timer creation and removal are expressed via $\text{StartTimer}(\cdot)$ and $\text{DeleteTimers}(\cdot)$ in the transition rules.

4.5.3. Radio Layer Constraints (Level 0)

Level 0 rules enforce radio-layer constraints that can be evaluated without any protocol history. These checks validate whether incoming frames comply with region-specific frequency plans and data rate assignments defined by the LoRaWAN specification. Since such violations are physically implausible or configuration-inconsistent, they can be detected early and rejected with minimal computational cost.

The first rule in Equation 4 illustrates well the general structure used throughout the specification. It should be read as follows: if an incoming event e corresponds to a `Join Request` message, and the device is in a state where a join request is admissible, and the message is received using a data rate different from the one the message is supposed to use according to the LoRaWAN specification, then the event is rejected and the system state remains unchanged. All subsequent rules follow the same guard–action pattern, differing only in the specific conditions and actions applied.

$$
\begin{aligned}
&(e \in \mathcal{E}_{\text{msg}} \wedge m(e).\text{type} = \text{JR} \wedge \mathit{state} \in \text{CAN\_REQUEST\_JOIN} \wedge m(e).\text{dr} \neq DR_{\text{JR}}) \\
&\quad \Rightarrow (v, e) \xrightarrow{\text{Reject\_JR\_FREQ}} v \\
&(e \in \mathcal{E}_{\text{msg}} \wedge m(e).\text{type} = \text{JR} \wedge \mathit{state} \in \text{CAN\_REQUEST\_JOIN} \wedge m(e).\text{freq} \neq F_{\text{JR}}) \\
&\quad \Rightarrow (v, e) \xrightarrow{\text{Reject\_JR\_FREQ}} v \\
&(e \in \mathcal{E}_{\text{msg}} \wedge m(e).\text{type} = \text{JA} \wedge \mathit{state} = \text{JOINING\_RX1} \wedge m(e).\text{freq} \neq F_{\text{Up}}) \\
&\quad \Rightarrow (v, e) \xrightarrow{\text{Reject\_JA\_RX1\_FREQ}} v \\
&(e \in \mathcal{E}_{\text{msg}} \wedge m(e).\text{type} = \text{JA} \wedge \mathit{state} = \text{JOINING\_RX1} \wedge m(e).\text{dr} \neq DR_{\text{Up}}) \\
&\quad \Rightarrow (v, e) \xrightarrow{\text{Reject\_JA\_RX1\_DR}} v \\
&(e \in \mathcal{E}_{\text{msg}} \wedge m(e).\text{type} = \text{JA} \wedge \mathit{state} = \text{JOINING\_RX2} \wedge m(e).\text{freq} \neq F_{\text{JA\_RX2}}) \\
&\quad \Rightarrow (v, e) \xrightarrow{\text{Reject\_JA\_RX2\_FREQ}} v \\
&(e \in \mathcal{E}_{\text{msg}} \wedge m(e).\text{type} = \text{JA} \wedge \mathit{state} = \text{JOINING\_RX2} \wedge m(e).\text{dr} \neq DR_{\text{JA\_RX2}}) \\
&\quad \Rightarrow (v, e) \xrightarrow{\text{Reject\_JA\_RX2\_DR}} v
\end{aligned}
\tag{4}
$$

Together, these rules ensure that only radio-compliant frames are forwarded to higher validation levels, preventing malformed or implausible traffic from influencing protocol state.

4.5.4. Flow Guard Constraints (Level 1)

Level 1 flow guard rules, presneted in Equation 5 ensure that messages are processed only in semantically valid protocol states. These constraints prevent invalid message ordering, such as join requests or join-accept messages appearing outside their designated protocol phases.

$$
\begin{aligned}
&(e \in \mathcal{E}_{\text{msg}} \wedge m(e).\text{type} = \text{JR} \wedge \mathit{state} \notin \text{CAN\_REQUEST\_JOIN}) \\
&\quad \Rightarrow (v, e) \xrightarrow{\text{Reject\_JR\_STATE}} v \\
&(e \in \mathcal{E}_{\text{msg}} \wedge m(e).\text{type} = \text{JA} \wedge \mathit{state} \notin \text{CAN\_ACCEPT\_JOIN}) \\
&\quad \Rightarrow (v, e) \xrightarrow{\text{Reject\_JA\_STATE}} v
\end{aligned}
\tag{5}
$$

 doi:10.20944/preprints202602.0072.v1

By enforcing state-dependent admissibility, these rules protect the Digital Twin from premature, delayed, or out-of-context protocol actions.

4.5.5. Security Constraints (Level 2)

Level 2 rules (see Equation 6 introduce security-oriented guardrails that leverage historical protocol context. These checks implement replay protection and cryptographic integrity validation by correlating incoming messages with previously observed events, such as DevNonce reuse or inconsistent gateway reception patterns.

$$\begin{aligned}
&(e \in \mathcal{E}_{\text{msg}} \land m(e).\text{type} = \text{JR} \land \text{DevNoncePresent}(m(e)) \land \text{ReplaySameGW}(m(e))) \\
&\quad \Rightarrow (v, e) \xrightarrow{\text{Reject_JR_REPLAY}} v \\
&(e \in \mathcal{E}_{\text{msg}} \land m(e).\text{type} = \text{JR} \land \text{DevNoncePresent}(m(e)) \land \text{SeenFromOtherGW}(m(e))) \\
&\quad \Rightarrow (v, e) \xrightarrow{\text{Info_JR_DEVNONCE}} v \\
&(e \in \mathcal{E}_{\text{msg}} \land m(e).\text{type} = \text{JR} \land \text{MICInvalid}(m(e))) \\
&\quad \Rightarrow (v, e) \xrightarrow{\text{Reject_JR_MIC}} v \\
&(e \in \mathcal{E}_{\text{msg}} \land m(e).\text{type} = \text{JA} \land \text{MICInvalid}(m(e))) \\
&\quad \Rightarrow (v, e) \xrightarrow{\text{Reject_JA_MIC}} v
\end{aligned} \tag{6}$$

Unlike purely statistical replay detection, these rules explicitly encode protocol-defined freshness guarantees, enabling precise identification of security violations during the activation phase.

4.5.6. State Transition Flow (Level 3)

Level 3 rules (see Equations 7, 8, 9, 10, 11, 12, 13) define the functional progression of the OTAA procedure itself. These rules are responsible for updating protocol state, managing timers associated with reception windows, and establishing session context upon successful activation.

$$\begin{aligned}
&(e \in \mathcal{E}_{\text{msg}} \land m(e).\text{type} = \text{JR} \land \mathit{state} \in \text{CAN_REQUEST_JOIN} \land \text{MICValid}(m(e))) \\
&\quad \Rightarrow (v, e) \xrightarrow{\text{JR_Allowed}} v' \\
&\quad \text{where } v' = \langle \text{JOINING_RX1DELAY}, \mathit{history}' \rangle \\
&\quad \land \mathit{history}' = \text{SaveDevNonce}(\mathit{history}, m(e)) \\
&\quad \land \text{StartTimer}(\text{JOINING}, \text{JA_TOUT_RX1START}) \\
&\quad \land \text{StartTimer}(\text{JOINING}, \text{JA_TOUT_RX1END}) \\
&\quad \land \text{StartTimer}(\text{JOINING}, \text{JA_TOUT_RX2START}) \\
&\quad \land \text{StartTimer}(\text{JOINING}, \text{JA_TOUT_RX2END})
\end{aligned} \tag{7}$$

$$\begin{aligned}
&(e \in \mathcal{E}_{\text{tmr}} \land \text{Expired}(\text{JA_TOUT_RX1START}) \land \mathit{state} = \text{JOINING_RX1DELAY}) \\
&\quad \Rightarrow (v, e) \xrightarrow{\text{RX1_START}} \langle \text{JOINING_RX1}, \mathit{history} \rangle
\end{aligned} \tag{8}$$

$$\begin{aligned}
&(e \in \mathcal{E}_{\text{tmr}} \land \text{Expired}(\text{JA_TOUT_RX1END}) \land \mathit{state} = \text{JOINING_RX1}) \\
&\quad \Rightarrow (v, e) \xrightarrow{\text{RX1_END}} \langle \text{JOINING_RX2DELAY}, \mathit{history} \rangle
\end{aligned} \tag{9}$$

$$\begin{aligned}
&(e \in \mathcal{E}_{\text{tmr}} \land \text{Expired}(\text{JA_TOUT_RX2START}) \land \mathit{state} = \text{JOINING_RX2DELAY}) \\
&\quad \Rightarrow (v, e) \xrightarrow{\text{RX2_START}} \langle \text{JOINING_RX2}, \mathit{history} \rangle
\end{aligned} \tag{10}$$

$$\begin{aligned}
&(e \in \mathcal{E}_{\text{tmr}} \land \text{Expired}(\text{JOINACCEPT_TOUT_RX2END}) \land \mathit{state} = \text{JOINING_RX2}) \\
&\quad \Rightarrow (v, e) \xrightarrow{\text{Reject_JA_TIMEOUT}} \langle \text{NDEF}, \mathit{history} \rangle
\end{aligned} \tag{11}$$

 doi:10.20944/preprints202602.0072.v1

$$
\begin{aligned}
&(e \in \mathcal{E}_{\text{msg}} \land m(e).\text{type} = \text{JA} \land \mathit{state} \in \text{CAN\_ACCEPT\_JOIN} \land \text{MICValid}(m(e))) \\
&\quad \Rightarrow (v, e) \xrightarrow{\text{Accept\_JA}} v' \\
&\quad \text{where } v' = \langle \text{JOINED},\ \mathit{history}' \rangle \\
&\quad \land\ \mathit{history}' = \text{SaveJoinAccept}(\mathit{history}, m(e)) \\
&\quad \land\ \mathit{history}' = \text{DeriveSessionKeys}(\mathit{history}') \\
&\quad \land\ \text{DeleteTimers}(\text{JOINING})
\end{aligned}
\tag{12}
$$

$$
\begin{aligned}
&(e \in \mathcal{E}_{\text{msg}} \land m(e).\text{type} = \text{DATA} \land \mathit{state} = \text{JOINED}) \\
&\quad \Rightarrow (v, e) \xrightarrow{\text{Valid\_DATA}} v
\end{aligned}
\tag{13}
$$

This explicit modeling of timing windows and state transitions allows the Digital Twin to detect subtle violations of protocol behavior, such as delayed join-accept messages or desynchronized reception attempts, which would remain invisible to packet-centric detectors.

The generic fallback rule included in the implementation ensures total event handling by the rule engine but does not represent protocol behavior and is therefore excluded from the formal specification of the join procedure.

$$
\begin{aligned}
&(e \in \mathcal{E}) \\
&\quad \Rightarrow (v, e) \xrightarrow{\text{Reject\_UNHANDLED}} v
\end{aligned}
\tag{14}
$$

For improved interpretability, a simplified set of states and transitions defined by the above rules is summarized in a graphical finite state machine representation in Figure 2. Blue transitions denote message-triggered events, while red transitions correspond to timer-triggered state changes. The diagram illustrates both nominal protocol progression and timing-driven deviations across RX1 and RX2 reception windows, including recovery paths and terminal failure states.

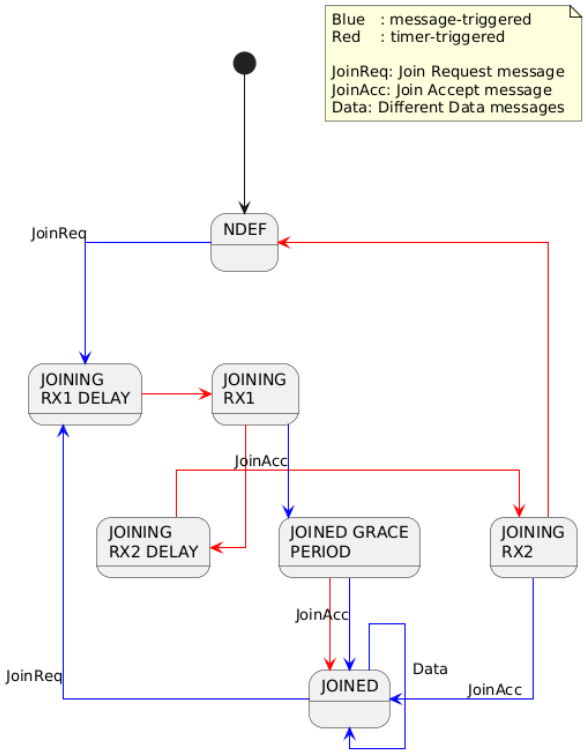

Figure 2. Finite state machine representation of the LoRaWAN OTAA procedure as implemented by the Digital Twin–based intrusion detection system. Blue transitions denote message-triggered events, while red transitions correspond to timer-triggered state changes associated with RX1 and RX2 reception windows.

 doi:10.20944/preprints202602.0072.v1

5. Experimental Setup and System Implementation

This section describes the experimental environment and system implementation used to realize the protocol-aware Digital Twin–based intrusion detection framework introduced in Section 4. The objective of the implementation is twofold: (i) to provide a realistic, reproducible LoRaWAN deployment that reflects operational constraints, and (ii) to demonstrate how the formally specified FSM-based detection logic can be embedded into a practical, SOC-integrated monitoring pipeline.

5.1. LoRaWAN Testbed Environment

The experimental setup is based on a physical LoRaWAN network operated at Obuda University, complemented by cloud-hosted backend components. The testbed mirrors a typical real-world deployment in which resource-constrained end devices communicate with a centralized network server through multiple gateways, while security monitoring and analysis are performed at backend and SOC levels.

The LoRaWAN infrastructure consists of two RAK7246G WisGate Developer D0 gateways, each built on Raspberry Pi hardware and equipped with Semtech SX1308 concentrators. The gateways forward uplink traffic to an open-source ChirpStack LoRaWAN network server [25] deployed on virtual machines within the T-Systems Open Telekom Cloud (OTC). Backend services required by ChirpStack, including PostgreSQL, Redis, and an MQTT broker (Eclipse Mosquitto), are hosted within the same cloud environment.

End-device traffic is generated by commercial off-the-shelf LoRaWAN sensors (Seeed Studio SenseCAP S2101 and S2102), which periodically transmit environmental measurements such as temperature, humidity, light intensity, and battery level. In addition, programmable Wio-E5-based LoRaWAN clients are connected to Raspberry Pi boards via serial interfaces, allowing controlled experimentation with join procedures, key configurations, and device registration states.

The testbed operates exclusively on LoRaWAN version 1.0.3, reflecting its widespread adoption in production deployments and its default support in ChirpStack. This choice ensures both practical relevance and reproducibility.

5.2. Monitoring Points and Data Acquisition

To support protocol-aware intrusion detection, LoRaWAN communication is observed at multiple points along the communication pipeline as depicted in Figure 3. These monitoring points correspond to locations where protocol metadata, timing information, or session context can be accessed without interfering with normal operation.

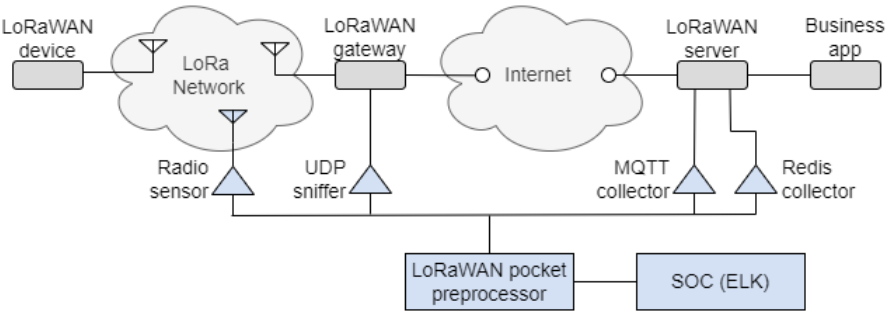

Figure 3. Main components of the LoRaWAN experimental test environment, illustrating end devices, gateways, backend services, and monitoring points (denoted with triangles) used for protocol-aware intrusion detection. The setup reflects a realistic deployment in which protocol metadata is collected from multiple layers without interfering with normal network operation.

In the current implementation, two monitoring mechanisms are fully operational beyond proof-of-concept level:

 doi:10.20944/preprints202602.0072.v1

- **MQTT-based monitoring**: decoded telemetry published by the gateway forwarders and network server is captured by subscribing to relevant MQTT topics. Messages encoded in Protobuf format are decoded and transformed into structured JSON records.
- **Redis-based frame log access**: internal frame logs maintained by the ChirpStack network server are retrieved from Redis streams using authenticated API access. These logs provide detailed protocol-level context, including message types, device identifiers, and session-related metadata.

Additional monitoring approaches, such as SDR-based radio capture and UDP interception between gateways and forwarders, are available in proof-of-concept form and are not required for the core operation of the Digital Twin–based IDS. All collected telemetry is forwarded to the preprocessing layer for normalization and enrichment.

5.3. Preprocessing Pipeline and FSM Integration

All LoRaWAN telemetry passes through a dedicated preprocessing layer prior to ingestion into the SIEM. This layer is responsible for protocol normalization, metadata enrichment, and execution of the FSM-based Digital Twin described in Section 4.5.

The preprocessing pipeline is implemented as a modular, Python-based service that processes incoming events from heterogeneous sources (MQTT, Redis) in a unified manner. Events are transformed into a common internal representation that exposes protocol-relevant fields such as message type, timing metadata, radio parameters, and device identifiers.

The FSM-based Digital Twin is embedded directly within this preprocessing layer and operates as an executable specification. Incoming events are evaluated against the hierarchical rule system (Levels 0–3), and protocol state is maintained on a per-device basis. State transitions, timer scheduling, and security violations are handled locally within the preprocessor, ensuring that protocol semantics are enforced before events reach the SIEM layer. The sequence diagram of the rule-based Digital Twin depicted in Figure 4.

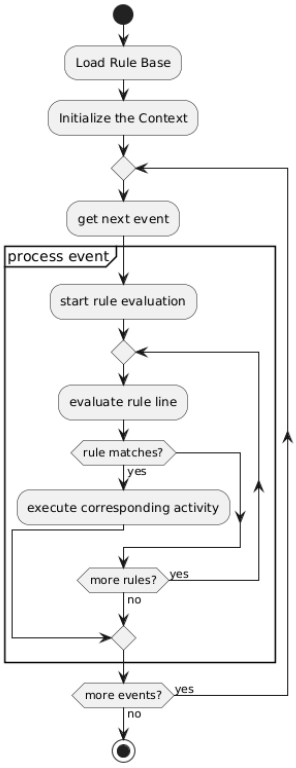

Figure 4. Sequence diagram illustrating the internal operation of the rule-based FSM engine embedded in the preprocessing layer. Incoming events are collected, evaluated against hierarchical protocol rules, used to update the communication context, and emitted as structured detection records toward the SIEM layer.

 doi:10.20944/preprints202602.0072.v1

This design allows the preprocessing layer to operate in Cryptography-assisted mode, where access to AppKeys enables session key derivation, MIC validation, and payload decryption, allowing full protocol field analysis.

Figure 5 presents the end-to-end operation of the preprocessing pipeline, highlighting how protocol-aware detection is integrated before SIEM ingestion. Telemetry streams collected from gateways and backend components are ingested by the preprocessing service, where events are decoded, enriched, and mapped into a unified internal representation. The embedded FSM-based Digital Twin evaluates each event in sequence, maintaining per-device protocol state and applying the hierarchical rule set. Detection outcomes, including state transitions, rule violations, and timing-related events, are emitted as structured records and forwarded to the SIEM layer. By performing protocol validation prior to SIEM ingestion, the pipeline ensures that downstream security analytics operate on semantically enriched, state-aware events rather than raw or decontextualized logs. This design preserves protocol semantics while remaining compatible with standard SOC workflows.

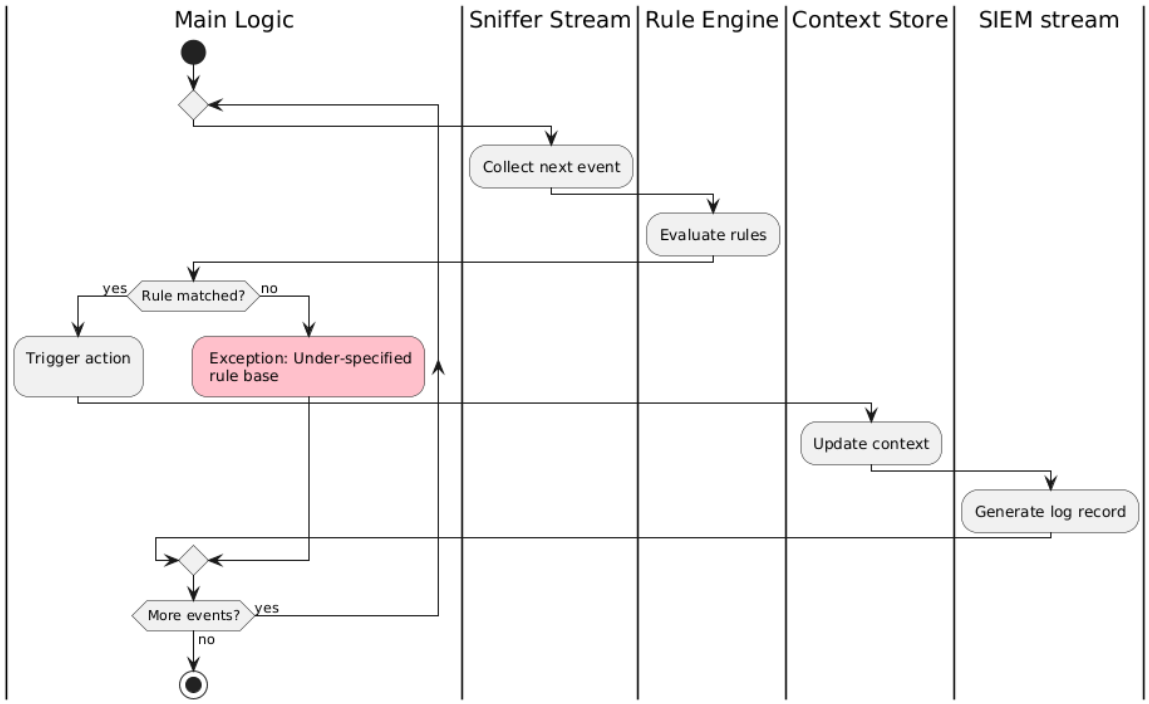

Figure 5. End-to-end preprocessing pipeline integrating protocol-aware Digital Twin–based detection into SOC workflows. Telemetry streams from gateways and backend components are decoded, normalized, and enriched before being evaluated by the FSM. Detection outcomes, including state transitions and rule violations, are forwarded to the SIEM as semantically enriched events.

5.4. SIEM Integration and Visualization

Processed events and detection outcomes generated by the preprocessing layer are forwarded to an ELK-based [26] SIEM infrastructure. Elasticsearch is used for storage and indexing, while Kibana provides dashboards and alerting mechanisms for SOC analysts.

FSM state transitions, rule violations, and contextual metadata are mapped to structured fields, enabling correlation across devices, gateways, and time windows. This integration allows protocol-level anomalies, such as invalid join sequences, nonce reuse, timing violations, or inconsistent gateway metadata, to be visualized and acted upon within standard SOC workflows.

Importantly, the SIEM layer remains decoupled from protocol logic. All protocol-aware reasoning is confined to the preprocessing layer, while the SIEM focuses on aggregation, visualization, and alert management. This separation of concerns improves maintainability and allows the Digital Twin logic to evolve independently of the SOC tooling.

 doi:10.20944/preprints202602.0072.v1

6. Results

To demonstrate the operational behavior of the proposed Digital Twin–based IDS and its SIEM-facing outputs, we executed a set of controlled OTAA interaction scenarios in our LoRaWAN test environment. The scenarios were designed to cover (i) nominal protocol execution, and (ii) representative deviations that stress different validation layers of the hierarchical rule engine (radio constraints, protocol flow guarding, security checks, and timing/state progression). Each experiment generates a stream of normalized events that is processed by the preprocessing layer where the FSM-based Digital Twin maintains per-device protocol state and emits structured detection outcomes before SIEM ingestion.

Table 3. OTAA protocol compliance and anomaly detection scenarios evaluated using the Digital Twin–based IDS.

ID	Scenario description	Deviation type	FSM final state	Detection level	SIEM outcome
S1	Join-Request → Join-Accept → Data	None (baseline)	JOINED	–	No alert
S2	Join-Request with reused DevNonce observed from another gateway, followed by Join-Accept and Data	DevNonce reuse (multi-GW)	JOINED	L2 (Sec)	Notice
S3	Join-Request with invalid MIC	MIC integrity violation	NDEF	L2 (Sec)	Reject
S4	Join-Request transmitted on invalid frequency	Radio parameter violation	NDEF	L0 (Radio)	Reject
S5	Join-Request → timeout → new Join-Request (new DevNonce) → Join-Accept → Data	Timing deviation with recovery	JOINED	L3 (Flow)	Notice
S6	Join-Request → Join-Accept received only in RX2 window	RX1 window miss	JOINED	L3 (Timing)	Notice
S7	Join-Request → Join-Accept received after RX2 window	RX window violation	NDEF	L3 (Timing)	Reject
S8	Join-Request → timeout → repeated Join-Request with identical DevNonce	Replay after timeout	NDEF	L2 (Sec)	Reject
S9	Join-Request → Join-Accept with invalid MIC	MIC integrity violation	NDEF	L2 (Sec)	Reject

As summarized in Table 3, the evaluated scenarios cover both nominal protocol execution (S1) and representative deviations involving nonce reuse (S2, S8), timing (S5, S6, S7), radio parameter violations (S4), and cryptographic integrity failures (S3, S9). For each scenario, the Digital Twin–based IDS produces a deterministic FSM outcome and a corresponding SIEM-level alert classification.

6.1. FSM State Evolution and Detection Outcomes

Figure 6 illustrates a representative excerpt from the SIEM event stream generated by the FSM-based preprocessing layer during OTAA execution. Each entry corresponds to a protocol-relevant event processed by the Digital Twin.

The fields prevState and newState capture the Digital Twin's current protocol context and its evolution, while MSG.Type indicates whether the triggering event is a protocol message (e.g., JOIN_REQUEST, JOIN_ACCEPT, UNCONFIRMED_DATA_UP) or an internal timer event (shown as null in the message type field). The action field provides a human-readable explanation of the decision (e.g., Valid `JoinRequest`, `Join Accept` RX1 timeout, Duplicate DevNonce from other GW, MIC invalid), and the icons in the 3rd column encodes the alert severity (e.g., informational/notice vs. reject). This visualization demonstrates a key operational property: the IDS is not limited to "packet verdicts", but produces explainable, stateful protocol reasoning that is directly consumable by SOC workflows.

The first sequence (lines 1-5) in the figure demonstrates a failed join attempt caused by a Join-Accept timeout. Following a valid Join-Request, the FSM transitions through the RX1 and RX2 reception windows. As no valid Join-Accept is observed within the protocol-defined timing constraints, the

 doi:10.20944/preprints202602.0072.v1

#	timestamp	...	action	event	MSG type	prevState	newState
1	16:25:41.482	✓	Valid JoinRequest	Message	JOIN_REQUEST	NDEF	JOINING_RX1DELAY
2	16:25:46.442	✓	Starting RX1 window	Timer	(null)	JOINING_RX1DELAY	JOINING_RX1
3	16:25:47.432	i	Join Accept RX1 timeout	Timer	(null)	JOINING_RX1	JOINING_RX2DELAY
4	16:25:47.442	✓	Starting RX2 window	Timer	(null)	JOINING_RX2DELAY	JOINING_RX2
5	16:25:48.432	✗	Join Accept timeout	Timer	(null)	JOINING_RX2	NDEF
6	16:25:50.482	✓	Valid JoinRequest	Message	JOIN_REQUEST	NDEF	JOINING_RX1DELAY
7	16:25:50.682	i	Duplicate DevNonce from other GW	Message	JOIN_REQUEST	JOINING_RX1DELAY	JOINING_RX1DELAY
8	16:25:55.442	✓	Starting RX1 window	Timer	(null)	JOINING_RX1DELAY	JOINING_RX1
9	16:25:55.482	✗	Data Message invalid state	Message	UNCONFIRMED_D...	JOINING_RX1	JOINING_RX1
10	16:25:55.782	✓	Valid JoinAccept in First Rec Window	Message	JOIN_ACCEPT	JOINING_RX1	JOINED_GRACE_P
11	16:25:56.782	✓	Second Join Accept during the Grace P...	Message	JOIN_ACCEPT	JOINED_GRACE_P	JOINED
12	16:25:58.482	✗	MIC invalid	Message	JOIN_REQUEST	JOINED	JOINED
13	16:26:00.482	✓	Valid JoinRequest	Message	JOIN_REQUEST	JOINED	JOINING_RX1DELAY
14	16:26:05.442	✓	Starting RX1 window	Timer	(null)	JOINING_RX1DELAY	JOINING_RX1
15	16:26:06.432	i	Join Accept RX1 timeout	Timer	(null)	JOINING_RX1	JOINING_RX2DELAY
16	16:26:06.442	✓	Starting RX2 window	Timer	(null)	JOINING_RX2DELAY	JOINING_RX2
17	16:26:06.782	✓	Valid JoinAccept in Second Rec Window	Message	JOIN_ACCEPT	JOINING_RX2	JOINED
18	16:26:08.482	✓	Valid JoinRequest	Message	JOIN_REQUEST	JOINED	JOINING_RX1DELAY
19	16:26:13.442	✓	Starting RX1 window	Timer	(null)	JOINING_RX1DELAY	JOINING_RX1
20	16:26:14.432	i	Join Accept RX1 timeout	Timer	(null)	JOINING_RX1	JOINING_RX2DELAY
21	16:26:14.442	✓	Starting RX2 window	Timer	(null)	JOINING_RX2DELAY	JOINING_RX2
22	16:26:14.782	✗	JoinAccept Frequency not allowed	Message	JOIN_ACCEPT	JOINING_RX2	JOINING_RX2
23	16:26:15.432	✗	Join Accept timeout	Timer	(null)	JOINING_RX2	NDEF

Figure 6. Kibana-based SIEM visualization of FSM evaluation results during OTAA execution. Each log entry corresponds to a protocol-relevant event processed by the Digital Twin, showing previous and new FSM states, triggering message or timer events, and rule-level decisions. The visualization demonstrates how protocol semantics are exposed directly to SOC analysts.

Digital Twin triggers a timeout rule at the end of the RX2 window, transitioning the FSM into the terminal `NDEF` state. This behavior corresponds to scenario S7 in Table 1 and illustrates how timing violations are deterministically enforced by the protocol model.

6.2. Handling of Benign Deviations and Recovery

The subsequent sequence (lines 6-11) demonstrates a recovery scenario in which a new Join-Request is issued after a failed activation attempt. When a valid Join-Request with a fresh DevNonce is received, the FSM correctly re-enters the join procedure and transitions through the RX1 and RX2 windows. Upon successful reception of a valid Join-Accept message, the FSM reaches the `JOINED` state, after which normal data traffic is accepted.

6.3. Detection of Suspicious but Tolerated Behavior

Figure 6 also highlights a scenario in which a duplicate DevNonce is observed from a different gateway during the join procedure (line 7). In this case, the Digital Twin raises an informational notice while maintaining the current FSM state, allowing the join procedure to continue. This behavior corresponds to scenario S2 in Table 3 and reflects a deliberate design choice: the IDS distinguishes between protocol violations that require immediate rejection and suspicious conditions that warrant monitoring without disrupting legitimate device activation.

6.4. Cryptographic and Protocol-Level Enforcement

Cryptographic integrity violations are consistently enforced at the security validation level. As shown in the log excerpt, Join-Request messages with invalid MIC values (line 12) are rejected without triggering state transitions, ensuring that malformed or forged messages cannot influence protocol state. Similar enforcement applies to Join-Accept messages failing integrity verification, as reflected by transitions to the `NDEF` state in scenarios S3 and S9.

 doi:10.20944/preprints202602.0072.v1

6.5. SIEM-Level Interpretability

Importantly, all detection outcomes generated by the Digital Twin are propagated to the SIEM layer as structured, interpretable events. Each alert includes the FSM state, the triggered rule level, and the specific rule identifier responsible for the decision. This enables SOC analysts to trace alerts back to protocol semantics rather than opaque anomaly scores, improving explainability and operational trust.

Beyond per-event explanations, the structured fields enable aggregation and dashboarding: analysts can query, for example, all Level 2 integrity failures, or all RX-window-related notices, and correlate them with gateway context and radio metadata. This makes protocol violations measurable at SOC scale and supports both real-time alerting and retrospective incident analysis.

6.6. FSM-Based End-Device State Evolution

Figure 7 provides a complementary, device-centric view of the Digital Twin–based intrusion detection process by visualizing the state evolution of an end device throughout successive OTAA interactions. While the SIEM excerpts in Figure 6 focus on backend-level event streams and rule evaluations, this figure reconstructs the logical protocol trajectory as maintained by the FSM for a single device instance.

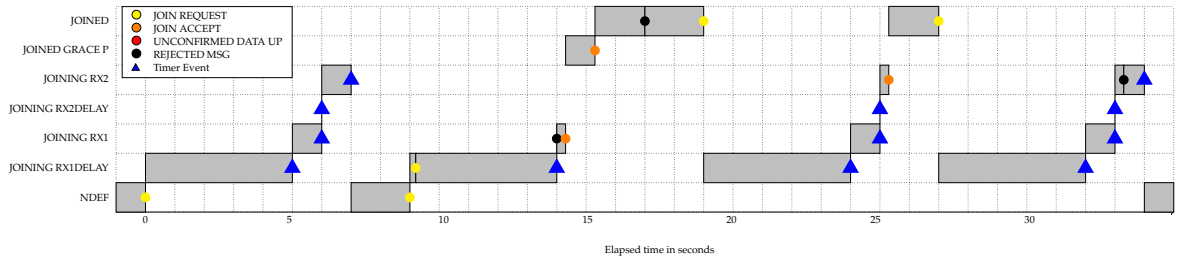

Figure 7. Temporal evolution of an end device's protocol state as maintained by the FSM-based Digital Twin during successive OTAA interactions. State transitions are driven by both message arrivals and timer expirations, illustrating failed join attempts, recovery through fresh Join Requests, and successful activation.

The visualization highlights how the Digital Twin transitions deterministically between protocol states (e.g., NDEF, JOINING_RX1, JOINING_RX2, JOINED) in response to both message arrivals and timer expirations. Failed activation attempts, recovery through fresh join requests, and successful session establishment are represented as explicit state transitions rather than implicit side effects of packet handling.

This representation makes protocol-level deviations, such as prolonged residence in joining states, repeated activation attempts, or timing-driven fallbacks directly observable. By exposing the end-device lifecycle as a sequence of well-defined states, the figure illustrates a key advantage of the proposed approach: protocol misuse and anomalous behavior are not inferred indirectly from traffic statistics, but are identified through violations of expected state progression. This device-centric perspective reinforces the interpretability of the Digital Twin model and complements the SIEM-oriented results by linking backend detection outcomes to concrete protocol behavior.

7. Discussion

This work demonstrates that protocol-aware intrusion detection for LoRaWAN can be realized as an operationally deployable Digital Twin that is both state-synchronized and SIEM-native. In contrast to detectors that operate on isolated packets or on coarse correlation rules, the proposed approach continuously maintains the OTAA execution context and evaluates events against an executable specification, yielding decisions that are inherently explainable through explicit state transitions and rule-level attribution.

 doi:10.20944/preprints202602.0072.v1

7.1. What the Approach Adds Beyond Conventional LoRaWAN Monitoring

A recurring limitation in LoRaWAN SOC practice is that many security-relevant conditions are only meaningful when interpreted in the correct protocol phase. The presented results illustrate that anomalies such as RX-window violations, repeated join attempts, or suspicious DevNonce reuse cannot be reliably interpreted without a stateful model of expected progression. By embedding the FSM-based Digital Twin directly into the preprocessing pipeline (prior to SIEM ingestion), the system can attach protocol semantics to telemetry early, ensuring that downstream analytics do not operate on ambiguous, decontextualized events.

A second contribution is the layered decision structure (Levels 0–3), which enables operationally useful differentiation between (i) radio/configuration issues, (ii) protocol sequencing violations, (iii) security-integrity and replay signals, and (iv) timing-driven flow outcomes. This separation supports SOC triage and alert tuning: e.g., noisy radio-level misconfigurations can be handled differently from cryptographic integrity failures.

7.2. Relation to Our Prior Pipeline and to Existing Approaches

Compared to our earlier SOC integration work, which focused on identifying monitoring points and establishing a LoRaWAN-to-SIEM telemetry pipeline, the present solution moves the core detection logic from generic correlation into a formal, protocol-conformant behavioral model. The practical implication is that the SIEM no longer receives only enriched metadata, but also receives interpreted protocol events: explicit state changes, timer-driven transitions, and rule-level outcomes.

Relative to prior LoRaWAN IDS efforts that emphasize statistical features of join behavior (e.g., distributions of DevNonce patterns or join-request timing), the proposed method provides a complementary advantage: it directly encodes the expected OTAA semantics and produces deterministic, explainable outcomes. This is particularly relevant when the goal is not only detection accuracy, but also auditability, repeatability, and SOC-facing interpretability.

7.3. Limitations

The current evaluation focuses on the OTAA procedure and a representative set of deviations around join integrity, replay signals, and RX-window timing. While these cover a security-critical phase of LoRaWAN, the full attack surface includes post-join data-plane behaviors (FCnt anomalies, ADR misuse, MAC command abuse, downlink scheduling patterns, and device-class-specific timing). In addition, the present results are scenario-based; larger-scale deployment would require characterization of alert rates under benign churn (e.g., poor coverage causing repeated joins) and multi-device concurrency.

7.4. Future Directions

A natural extension of the proposed approach is to expand the FSM/EFSM state space beyond the OTAA procedure to cover the complete LoRaWAN device lifecycle, including data uplink and downlink exchanges, frame counter validation strategies, MAC command sequences, and class-dependent receive behavior. Such an extension would allow the SIEM to observe not only join correctness but also long-term protocol compliance, state drift, and subtle deviations emerging over prolonged operation.

Importantly, the methodology itself is not tightly coupled to a specific LoRaWAN specification version. While the present implementation targets LoRaWAN 1.0.3 to reflect widely deployed operational environments, the FSM-based Digital Twin can be adapted to LoRaWAN 1.1 with minimal structural modifications by incorporating rejoin procedures, Join Server interactions, and updated key derivation semantics. This highlights that the proposed model captures protocol logic at an abstract behavioral level rather than encoding version-specific message handling.

Beyond LoRaWAN, the presented Digital Twin abstraction can serve as a blueprint for protocol-aware intrusion detection in other LPWAN and IoT communication systems. Protocols such as NB-IoT, WirelessHART, or industrial IoT stacks with explicitly defined stateful handshakes and timing

 doi:10.20944/preprints202602.0072.v1

constraints can be modeled using the same FSM/EFSM principles, enabling interpretable, specification-driven security monitoring under constrained observability.

In addition, the Digital Twin framework supports the expression of security and operational policies that are difficult to capture using stateless rules. Incorporating regulatory and operational constraints—such as duty-cycle limitations, per-device transmission patterns, or channel occupancy policies—would enable detection of misbehaving devices and configuration errors that degrade network availability without necessarily violating cryptographic integrity.

Finally, the approach naturally lends itself to hybrid designs combining formal models with learning-based methods. The FSM provides a high-precision semantic baseline and produces structured outputs, including state trajectories, rule activations, and timing residuals, which can serve as explainable input features for anomaly scoring models without sacrificing interpretability.

Without an explicit state-aware model, scenarios such as delayed Join-Accept messages crossing RX-window boundaries or benign retry-based recovery (S5–S7 in Table 3) would remain indistinguishable from normal packet loss in SIEM-level logs, underscoring the necessity of protocol-aware Digital Twin modeling for reliable LoRaWAN security monitoring.

8. Conclusions

This paper presented a protocol-aware intrusion detection methodology for LoRaWAN networks based on a finite-state Digital Twin of the OTAA procedure. By formalizing the expected sequence of activation messages, timing constraints, and state transitions, the proposed approach enables continuous validation of observed communication against protocol semantics rather than relying on attack-specific signatures or purely statistical indicators.

Through a series of controlled OTAA scenarios executed in a realistic LoRaWAN testbed, we demonstrated that the Digital Twin based IDS produces deterministic, interpretable detection outcomes that can be directly integrated into SOC workflows. The results show that security relevant deviations, such as replay attempts, timing violations, and integrity failures which manifest naturally as violations of protocol state progression can be systematically detected when protocol state and cryptographic session context are available for analysis.

Compared to conventional LoRaWAN monitoring and prior IDS approaches, the key contribution of this work lies in shifting intrusion detection from packet-level anomaly spotting to executable protocol conformance checking. By embedding the FSM-based Digital Twin directly into the preprocessing layer, protocol semantics are attached to telemetry before SIEM ingestion, enabling explainable alerts grounded in explicit state transitions and rule evaluations.

While the present study focuses on the OTAA procedure, the methodology is intentionally generic and extensible. The same Digital Twin abstraction can be expanded to cover post-join data exchange, MAC command handling, and class-specific timing behavior, providing a unified framework for state-aware security monitoring across the full LoRaWAN device lifecycle.

This FSM-based Digital Twin can also be adapted to LoRaWAN 1.1 with limited structural modifications, more broadly, the presented approach provides a blueprint for protocol-aware intrusion detection in other LPWAN and IoT communication systems. Overall, the results highlight the practical and methodological advantages of Digital Twin driven, state-synchronized intrusion detection for securing protocol-centric IoT systems.

Author Contributions: Conceptualization, Zs.B., R.F., and E.K.; methodology, Zs.B., R.F., and E.K.; formal modeling, Zs.B., R.F., and E.K.; software, Zs.B.; investigation, Zs.B., R.F., and E.K.; writing, Zs.B., R.F., and E.K.; visualization, Zs.B. All authors have read and agreed to the published version of the manuscript.

Acknowledgments: This research was supported by access to the OTC cloud service provided by Deutsche Telekom TSI Hungary Ltd. We gratefully acknowledge the infrastructure support, which contributed to the results presented in this publication and supports the promotion of TSI OTC both nationally and internationally. During the preparation of this manuscript, the authors used ChatGPT (version 5.2) to support language refinement and

 doi:10.20944/preprints202602.0072.v1

improve textual clarity. The authors have reviewed and edited the output and take full responsibility for the content of this publication.

Code Availability: The reference implementation of the FSM-based Digital Twin and the associated preprocessing components is available as open-source software at https://github.com/zsoltbringye/LoRa_FSM, supporting reproducibility.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Tolvaly-Roşca, F.; Fekete, G.; Bíró, I.; Fekete, A.Z.; Forgó, Z. Real-Time IoT Solution to Monitor and Control dMVHR Units in Real-Life Environment. *Acta Polytechnica Hungarica* **2024**, *21*.
2. Pekarčík, P.; Chovancová, E.; Chovanec, M.; Štancel, M. A Centralized Approach to Intrusion Detection System Management: Design, Implementation and Evaluation. *Acta Polytechnica Hungarica* **2025**, *22*, 7–26.
3. Bringye, Z.; Fleiner, R.; Umhauser, B.; Aradi, M.; Kail, E. Extending SOC Capabilities to LoRaWAN: A Cloud-Integrated Intrusion Detection Framework for IoT Networks. *Acta Polytechnica Hungarica* **2026**, *23*.
4. Alliance®, L. LoRaWan Specification 1.0.3.
5. Semtech Corporation. LoRa Technology Overview. https://www.semtech.com/lora, n.d. Accessed: January 10, 2026.
6. Alliance®, L. LoRaWan Specification 1.1.
7. Butun, I.; Pereira, N.; Gidlund, M. Security Risk Analysis of LoRaWAN and Future Directions. *Future Internet* **2018**, *11*, 3. https://doi.org/10.3390/fi11010003.
8. Eldefrawy, M.; Butun, I.; Pereira, N.; Gidlund, M. Formal security analysis of LoRaWAN. *Computer Networks* **2019**, *148*, 328–339. https://doi.org/10.1016/j.comnet.2018.11.017.
9. Hessel, F.; Almon, L.; Hollick, M. LoRaWAN Security: An Evolvable Survey on Vulnerabilities, Attacks and their Systematic Mitigation. *ACM Transactions on Sensor Networks* **2023**, *18*. https://doi.org/10.1145/3561973.
10. Loukil, S.; Fourati, L.; Nayyar, A.; Chee, K.W.A. Analysis of LoRaWAN 1.0 and 1.1 Protocols Security Mechanisms. *Sensors* **2022**, *22*. https://doi.org/10.3390/s22103717.
11. Tomasin, S.; Zulian, S.; Vangelista, L. Security analysis of lorawan join procedure for internet of things networks. In Proceedings of the 2017 IEEE Wireless Communications and Networking Conference Workshops (WCNCW). IEEE, 2017, pp. 1–6.
12. Spadaccino, P.; Cuomo, F. Intrusion Detection Systems for IoT: opportunities and challenges offered by Edge Computing and Machine Learning. *arXiv preprint arXiv:2012.01174* **2022**. arXiv:2012.01174 [cs], https://doi.org/10.48550/arXiv.2012.01174.
13. Verzegnassi, E.G.M.; Tountas, K.; Pados, D.A.; Cuomo, F. Data conformity evaluation: a novel approach for IoT security. In Proceedings of the 2019 IEEE 5th World Forum on Internet of Things (WF-IoT). IEEE, 2019, pp. 842–846.
14. Liao, H.J.; Lin, C.H.R.; Lin, Y.C.; Tung, K.Y. Intrusion detection system: A comprehensive review. *Journal of network and computer applications* **2013**, *36*, 16–24.
15. Danish, S.M.; Nasir, A.; Qureshi, H.K.; Ashfaq, A.B.; Mumtaz, S.; Rodriguez, J. Network Intrusion Detection System for Jamming Attack in LoRaWAN Join Procedure. In Proceedings of the 2018 IEEE International Conference on Communications (ICC), 2018, p. 1–6. https://doi.org/10.1109/ICC.2018.8422721.
16. Horák, T.; Střelec, P.; Kováč, S.; Tanuška, P.; Nemlaha, E. Detection of IoT communication attacks on LoRaWAN gateway and server. In Proceedings of the Computer Science On-line Conference. Springer, 2023, pp. 489–497.
17. Fu, Y.; Yan, Z.; Cao, J.; Koné, O.; Cao, X. An Automata Based Intrusion Detection Method for Internet of Things. *Mobile Information Systems* **2017**, *2017*, 1–13. https://doi.org/10.1155/2017/1750637.
18. Le, A.; Loo, J.; Chai, K.; Aiash, M. A Specification-Based IDS for Detecting Attacks on RPL-Based Network Topology. *Information* **2016**, *7*, 25. https://doi.org/10.3390/info7020025.
19. Fuller, A.; Fan, Z.; Day, C.; Barlow, C. Digital twin: enabling technologies, challenges and open research. *IEEE access* **2020**, *8*, 108952–108971.
20. Eckhart, M.; Ekelhart, A. Towards security-aware virtual environments for digital twins. In Proceedings of the Proceedings of the 4th ACM workshop on cyber-physical system security, 2018, pp. 61–72.
21. Homaei, M.; Morales, V.G.; Gutierrez, O.M.; Gomez, R.M.; Caro, A. Smart Water Security with AI and Blockchain-Enhanced Digital Twins. *arXiv preprint arXiv:2504.20275* **2025**. arXiv:2504.20275 [cs], https://doi.org/10.48550/arXiv.2504.20275.

 doi:10.20944/preprints202602.0072.v1

22. El-Hajj, M. Leveraging Digital Twins and Intrusion Detection Systems for Enhanced Security in IoT-Based Smart City Infrastructures. *Electronics* **2024**, *13*, 3941. https://doi.org/10.3390/electronics13193941.
23. Schösser, A.; Khursheed, D.; Schulz, P.; Burmeister, F.; Fettweis, G. *Digital Twin of the Radio Environment: A Novel Approach for Anomaly Detection in Wireless Networks*; IEEE, 2023. https://doi.org/10.1109/GCWkshps58843.2023.10464447.
24. Bringye, Z. LoRa_FSM: Finite State Machine–Based Digital Twin for LoRaWAN OTAA. https://github.com/zsoltbringye/LoRa_FSM, 2025. Accessed: 2025-05-26.
25. ChirpStack. ChirpStack Open-Source LoRaWAN Network Server. https://www.chirpstack.io/, 2025. Accessed: May 26, 2025.
26. Elastic. Elastic Stack (ELK): Elasticsearch, Kibana & Logstash. https://www.elastic.co/elastic-stack, 2025. Accessed: May 26, 2025.