# Secure and Autonomic Framework for SCHC Context Management in LoRaWAN-IPv6 Networks

**Maryam Hatami**

**A Thesis**
**in**
**The Department**
**of**
**Computer Science and Software Engineering**

**Presented in Partial Fulfillment of the Requirements**
**for the Degree of**
**Master of Computer Science at**
**Concordia University**
**Montréal, Québec, Canada**

**August 2024**

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: Maryam Hatami

Entitled: Secure and Autonomic Framework for SCHC Context Management in LoRaWAN-IPv6 Networks

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

| | |
|---|---|
| Dr. Abdelhak Bentaleb | Chair |
| Dr. Abdelhak Bentaleb | Examiner |
| Dr. Yann-Gaël Guéhéneuc | Examiner |
| Dr. Sandra Cespedes | Thesis Supervisor(s) |
| Dr. John William Atwood | Thesis Supervisor(s) |

Approved by ______________________________

Dr. Charalambos Poullis Chair of Department or Graduate Program Director

Department of Computer Science and Software Engineering

Aug 27th, 2024

Dr. M. Debbabi, Dean of Faculty of Engineering and Computer Science

# Abstract

Secure and Autonomic Framework for SCHC Context Management in LoRaWAN-IPv6 Networks

Maryam Hatami

The rapid growth of IoT technologies demands seamless integration of communication protocols, especially for devices in constrained environments. This thesis focuses on integrating LoRaWAN with IPv6 networks by developing a secure and autonomous framework for managing SCHC (Static Context Header Compression) contexts. The research aims to create a secure, human-free solution that maintains reasonable data transmission times between LoRaWAN and IPv6, essential for widespread IoT adoption. For instance, in smart agriculture, dynamic SCHC updates support equipment mobility; in industrial automation, SCHC ensures consistent data transmission amid changing topologies; in smart cities, it adapts to evolving infrastructure; and in healthcare, it manages reliable health data transmission.

The thesis introduces the concept of SCHC Zero Context, enabling immediate device communication by embedding pre-configured SCHC rules during manufacturing. This innovation facilitates an efficient bootstrapping process. Additionally, a comparative analysis of BRSKI (Bootstrapping Remote Secure Key Infrastructure) and cBRSKI (constrained BRSKI) demonstrates the feasibility of the solution for energy-constrained IoT devices.

The research underscores the importance of integrating security across IoT architecture layers. By combining LoRaWAN's data link layer security with IPv6's global internet, the proposed framework offers robust protection against common threats, ensuring secure onboarding and communication throughout a device's life-cycle.

This work provides a scalable, secure solution for integrating LoRaWAN with IPv6, laying the groundwork for future research in secure and efficient dynamic SCHC context management.

# Acknowledgments

I would like to express my sincere gratitude to all those who have supported me throughout the course of my research and the writing of this thesis.

First and foremost, I would like to thank my supervisors, Dr. Cespedes and Dr. Atwood, for their continuous guidance, encouragement, and invaluable feedback. Their expertise and support have been crucial to the completion of this work.

This research was made possible through the support of Mitacs, under the Mitacs Accelerate program - project IT40720, and ANID - Basal Project FB0008. I am deeply grateful for the financial assistance and the opportunity to collaborate with industry partners, which significantly enhanced the practical impact of this research.

I am also thankful to my colleagues and friends, whose insights and encouragement were vital in overcoming the challenges faced during this journey.

Lastly, I would like to extend my deepest appreciation to my family for their unwavering support and belief in me, without which this thesis would not have been possible.

Thank you all.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

The concept of the Internet of Things (IoT) has been around since 1999, when it was first introduced by Kevin Ashton [3]. Since then, IoT has changed how we collect information and automate tasks. It has quickly expanded to include various technologies, particularly those that help connect many devices while using as little energy as possible. One type of technology that has become essential for IoT is Low-Power Wide Area Networks (LPWAN). These networks are designed to allow devices to communicate over long distances while using minimal power.

As more and more devices connect to the internet, it becomes increasingly important to have network systems that are both efficient and secure. IPv6 is a version of the internet protocol that provides enough address space for the millions of devices that make up the IoT. It also improves connectivity, allowing devices to communicate with each other more effectively. However, integrating IPv6 with LPWAN technologies like LoRaWAN presents some challenges, especially when it comes to ensuring that devices can manage their network settings securely and without human intervention.

One of the key challenges in IoT networks is managing the Static Context Header Compression (SCHC) context. SCHC is a protocol designed to compress data, making it more efficient to transmit over the constrained network resources typical of IoT environments. Standardized by the IETF (Internet Engineering Task Force), SCHC is now being expanded beyond just LPWAN technologies. However, as network configurations evolve, the SCHC context must be continuously updated to maintain efficient and secure communication. This thesis focuses on developing a secure and autonomic system for managing SCHC contexts in LoRaWAN-IPv6 networks to address these challenges.

## 1.1 Motivation

LPWANs are rapidly expanding, with the global market expected to exceed tens of billions of dollars by 2025, supporting billions of IoT devices. These networks are crucial for enabling large-scale IoT deployments in areas like smart cities and industrial automation.

The SCHC protocol is essential for bridging LPWANs with the broader IP ecosystem, allowing IP-based protocols like UDP, CoAP, and HTTP to operate efficiently in resource-constrained environments. SCHC achieves this by using predefined sets of rules, known as SCHC rules, which both the sender and receiver use to compress and decompress data packets. These rules specify how to remove redundant information from packet headers, significantly reducing the data that needs to be transmitted.

However, as IoT devices move across different network environments or as new communication protocols evolve, the SCHC context—which includes these compression rules—must be updated to ensure ongoing efficient and secure communication. For instance, a sensor moving from one LPWAN to another might need to update its IP address and adjust its SCHC rules to maintain seamless connectivity. Similarly, as new features or options are introduced in protocols like CoAP, the SCHC rules must be modified to support these changes.

The challenge is that the current SCHC standard relies on static contexts, which are fixed and cannot easily adapt to such changes. This limitation is particularly problematic in large-scale deployments, where IoT devices frequently switch between networks or need to accommodate new protocol features. In such scenarios, relying on manual updates is not only impractical but also unsustainable, leading to significant inefficiencies, increased operational costs, and potential security vulnerabilities.

The lack of a secure and autonomic method for managing SCHC context updates in dynamic environments is a significant challenge. This thesis addresses this gap by proposing several key innovations: the introduction of SCHC Zero Context for immediate device communication upon network connection, a comparative analysis of BRSKI and cBRSKI for autonomous onboarding, and the integration of security mechanisms across different layers of the IoT architecture. These contributions enable scalable, secure, and efficient SCHC context management for LoRaWAN-IPv6 networks, providing a comprehensive solution to the challenges of IoT communication in dynamic environments.

## 1.2 Thesis Organization

This thesis is organized into the following chapters:

- **Chapter 2: Background** - Provides basic information and explains the key ideas used in the research, setting the stage for the following chapters.
- **Chapter 3: Onboarding and Security Mechanisms** - Discusses how new devices connect to a network and the security measures involved.
- **Chapter 4: State of the Art** - Reviews current work in combining LPWAN and IPv6, focusing on solutions for connecting devices and relevant use cases.
- **Chapter 5: Problem Statement** - Describes the specific challenges that this research aims to solve.
- **Chapter 6: Proposed Solution** - Presents the solution designed to tackle the identified challenges.
- **Chapter 7: Description of the Solution** - Details how the solution works, including the system setup and main components.
- **Chapter 8: Results** - Discusses the outcomes of the research, including evaluations and comparisons with existing solutions.
- **Chapter 9: Conclusion and Future Work** - Summarizes the key findings of the research, discusses what they mean, and suggests areas for future research.

# Chapter 2

# Background

The Internet of Things (IoT) has transformed data collection by connecting devices like sensors and controllers to the Internet, enabling advancements in smart homes, healthcare, and industrial operations. However, the growing number of IoT devices makes manual deployment and management impractical, leading to potential human errors and difficulties in accessing some devices. To overcome these challenges, it's crucial for IoT devices to be deployed and managed autonomously, without human intervention. As these devices increasingly connect to the Internet using standards like IPv6, specific configurations are required to ensure they can communicate efficiently. Our research addresses these needs by enhancing the automatic deployment and management of IoT devices, focusing on making them more adaptable and easier to connect within an IPv6 environment. In this chapter, we introduce the relevant terminologies and technologies used in our proposed system.

## 2.1 IoT

The term Internet of Things (IoT) has been recognized for the past few years, gaining increased attention recently due to advancements in wireless technology. IoT empowers significant objects to perform tasks by facilitating communication via internet.

IoT significantly impacts domestic life, including assisted living, smart homes, and smart cars. In the business sector, it contributes to advancements in manufacturing and services, enhancing services, production, and quality. [4].

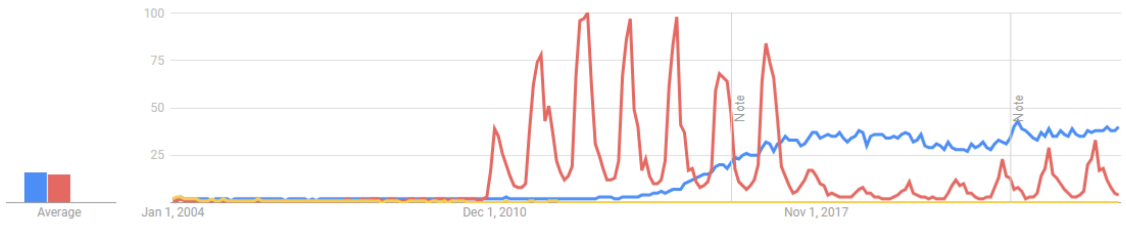


Figure 2.1: Google search trends since 2004 for terms Internet of Things, IoE, and ubiquitous computing. The relative value 0 to 100 represents search interest.

The popularity of various models including IoT devices changes over time. figure 2.1 illustrates the web search trends over the past 20 years for terms such as Internet of Things (indicated with blue), align with similar technologies such as IoE (The Internet of Everything) which is indicated by red, and ubiquitous computing (indicated by yellow), as measured by Google. Since the inception of IoT, there has been a consistent increase in search volume, contrasting with the declining trend for IoE in the recent years which includes human interaction and ubiquitous computing which requires more computing power. Google's search forecast suggests that this upward trend is likely to persist as other enabling technologies converge, ultimately forming a genuine Internet of Things.

## 2.2 LPWAN

Low-Power Wide Area Networks (LPWANs) are a class of wireless communication technologies which enable long-range communication between low power consumption devices. LPWANs has several distinguishing characteristics that set it apart from traditional communication networks:

- Low Power Consumption: Both the network and end devices are designed to use minimal energy.
- Cost-Effectiveness: Keeping communication, deployment, and management costs low is crucial, especially for large-scale deployments.
- Strong Security: A robust security mechanism is essential throughout the entire ecosystem, from devices to applications.
- Built-in Localization: Having built-in localization capabilities is advantageous, particularly for indoor deployments.
- Interference Resistance: In dense urban areas, network deployment can cause radio network congestion on the same or adjacent channels, necessitating robust modulation to resist interference.
- Efficient Data Handling: The data generated by nodes must be managed effectively.

## 2.3 LoRaWAN

Among the various LPWAN technologies, including LoRaWAN, SigFox, NB-IoT, Weightless, and other sub-GHz communication methods, LoRaWAN has gained significant attention from organizations and researchers, becoming a widely adopted LPWAN technology. LoRaWAN is better because it allows private network deployments, offers more flexible data policies than Sigfox, and operates independently of existing cellular networks like NB-IoT. Furthermore, despite the popularity of SigFox in the literature, its future is uncertain due to recent bankruptcy and acquisition [5]. Hence, it was intentionally excluded from our analysis. These make LoRaWAN more versatile, cost-effective, and suitable for a wider range of applications.

### 2.3.1 LoRa and LoRaWAN technologies

LoRa is a type of Radio Frequency (RF) modulation used at the physical layer of the OSI reference model. On the other hand, LoRaWAN is a standard for the medium access control (MAC) layer, which manages communication between devices. The LoRaWAN protocol brings following features for any devices using the standard:

- Channel management
- Energy efficiency
- Adaptive data rate
- Security

In this section, we will explore the technological aspects of both LoRa and LoRaWAN in depth.

### 2.3.2 LoRaWAN Architecture

A typical LoRaWAN network comprises the following components [6].:

- End Devices: These are sensors or actuators that send wireless messages modulated with LoRa to gateways. They can also receive messages wirelessly from gateways.
- Gateways: Gateways receive messages from end devices and then transmit them to the Network Server. Each gateway is registered to a LoRaWAN network server using configuration settings and receives LoRa messages from EDs, forwarding them seamlessly to the network server through various backhaul options such as Cellular (3G/4G/5G), WiFi, Ethernet, fiber-optic, or 2.4 GHz radio links.
- Network Server: This is a software component running on a server that manages the entire LoRaWAN network. The LoRa Network Server (NS) serves as the central manager for the entire LoRa network, overseeing gateways (GWs), end-devices (EDs), applications, and users. Its responsibilities encompass receiving up-link frames from gateways and relaying them to the corresponding Application Server (AS). The NS plays a crucial role in managing the Medium Access Control (MAC) layer, addressing tasks such as scheduling down-link data transmission, preventing packet duplication, coordinating acknowledgments, and adjusting data rates.
  Emphasizing security, the NS implements encryption at various layers, including NwKey encryption at the network layer, AppKey encryption at the application level, and encryption at the ED level using the device key.
  Additional features of a typical LoRaWAN Network Server include establishing secure 128-bit AES connections for end-to-end security, validating end device authenticity, deduplicating uplink messages, selecting optimal gateways for routing downlink messages, sending ADR commands to optimize device data rates, performing device address checking, providing acknowledgements for confirmed uplink data messages, forwarding uplink payloads to appropriate application servers, routing uplink payloads to the correct Application Server, forwarding Join-request and Join-accept messages between devices and join servers, and responding to all MAC layer commands.
- The Join Server is an additional software component running on a server which is responsible for overseeing the enrollment of End-devices (EDs) into the network and verifying their authenticity. Utilizing two distinct keys, JSIntKey for the Message Integrity Code (MIC) of the rejoin request message and the join accept answer, and JSEncKey for encrypting the join accept message, the Join Server plays a pivotal role in ensuring a secure device activation process, root key storage, and session key generation. The initiation of the join procedure involves the end device sending a Join-request message to the Join Server via the Network Server. The Join Server processes this message, generates session keys, and transmits NwkSKey and AppSKey to the Network Server and the Application Server, respectively. Introduced with LoRaWAN v1.1 and also available in LoRaWAN v1.0.4, the Join Server is an integral component in facilitating the secure integration of end devices into the LoRaWAN network, which will be discussed in detail later in the next chapter.

- Application Servers: These are software components running on servers responsible for securely processing application data. End-devices establish communication with the Application Server (AS) through a code or program operating on the end device itself. Acting as a remote entity, the AS manages the end devices actions and gathers information about the devices that are connected. This server processes application-specific data messages received from the end devices, generates application-layer downlink payloads, and transmits them to the connected end devices through the Network Server. Notably, a LoRaWAN network may accommodate multiple Application Servers.

Figure 2.2 illustrates the LoRaWAN architecture. This architecture uses a star-of-stars topology, centered around a Network Gateway that manages all data traffic between the devices and the non-constrained side of the network. These devices, which are the sensors and actuators, often use resource-limited, battery-powered microcontroller units (MCUs) and are deployed over large areas.

LoRa End-devices are linked to gateways. Each gateway connects to the network server (NS), which may link to multiple application servers (ASs). Messages transmitted by end devices propagate through all gateways within their coverage area. These messages are then collected by the Network Server. In instances where the Network Server receives multiple copies of the same message, it retains only one copy while discarding the duplicates. [7].

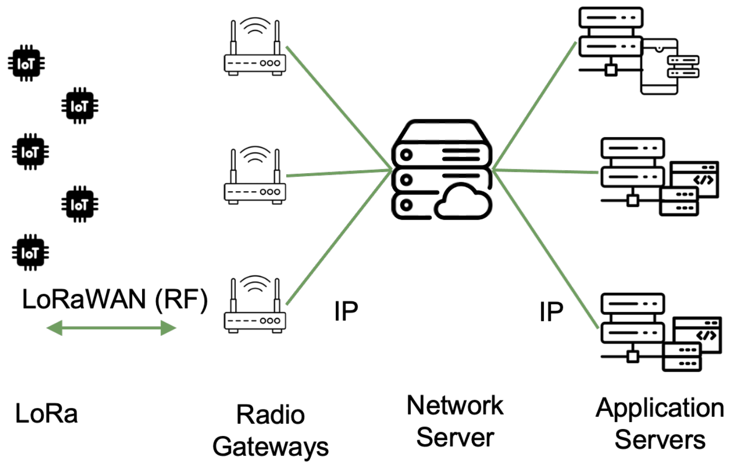


Figure 2.2: LoRaWAN Topology. (Source [8].)

### 2.3.3 LoRaWAN Protocol stack

The overall LoRaWAN protocol stack is structured into four layers, including Application layer, MAC Layer, Physical Layer, and Radio Frequency Layer. LoRa is a technology that operates at the Physical layer (L1) and is primarily used to transmit Application layer data over the communication medium. The OSI reference model's L2 Data Link Layer corresponds to the LoRaWAN protocol, which establishes secure medium access and manages end-nodes. Typically, end-nodes are designed to perform specific tasks to minimize energy consumption, thus they handle functionalities at the L1, L2, and L7 OSI layers. Radio Gateways (RGWs) serve as intermediaries between the end-nodes and the network server. The network server plays a crucial role in controlling the communication medium. LoRaWAN networks differ from traditional TCP/IP communication between gateways (GWs) and end-nodes. Network servers in LoRaWAN function as straightforward application services operating at the Transport Layer.

**LoRa Physical Layer**

Long Range (LoRa) represents a physical layer technology pioneered by Semtech [9], with the company holding intellectual property rights over the modulation. This modulation technique extends conventional Spread Spectrum principles, aiming to minimize the energy required for transmitting data over the channel. The achievable Data Rate (DR) in communication can be derived from parameters such as bandwidth (BW), Spreading Factor (SF), and Coding Rate (CR).

The Spreading Factor (SF) is a crucial parameter for maintaining quality of service in LoRaWAN. Lower SF values result in higher data rates and shorter air time, while higher SF values extend the communication range but reduce the quality of service. SF values from 7 to 12 enable orthogonal communication, meaning multiple networks can transmit simultaneously on the same frequency band without causing interference.

As depicted in the Physical layer in figure 2.3, The LoRa frame format offers two options: implicit and explicit. In the explicit packet format, a concise header is included, providing crucial details such as the number of bytes, CRC information, and the coding rate utilized within the frame.

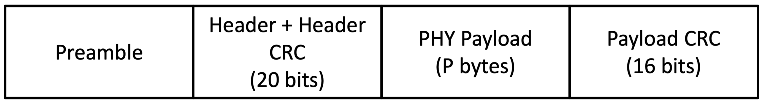


Figure 2.3: LoRa Frame. Source [7].

The relationship between the data transmission rate and the Spreading Factor (SF) is defined as follows. Equation 1 establishes the relationship between the Symbol Rate (RS), bandwidth (BW), and the spreading factor. The following equations are adapted from [10].

$$RS = \frac{BW}{2^{SF}} \tag{1}$$

The duration of frame airtime holds significant importance for real-time applications. Utilizing parameters such as CR, SF, and BW, it's possible to calculate the transmission time of a LoRa frame. This duration is the sum of the preamble $T_{\text{preamble}}$ transmission time and the payload transmission time $T_{\text{payload}}$ as specified in Equation (2).

$$T_{\text{frame}} = T_{\text{preamble}} + T_{\text{payload}} \tag{2}$$

The duration of the preamble, denoted as $T_{\text{preamble}}$ depends on the symbol time $T_{\text{sym}}$ given by Equation (3), and the programmable length of modem registers $n_{\text{preamble}}$ as in Equation (6). LoRaWAN 1.0 specifies the default preamble value as 8.

$$T_{\text{sym}} = \frac{1}{R_{\text{S}}} \tag{3}$$

$$T_{\text{preamble}} = (n_{\text{preamble}} + 4.25) \times T_{\text{sym}} \tag{4}$$

The time it takes to send a frame in LoRaWAN depends on the size of the payload $n_{\text{payload}}$, which is determined by several factors: Packet Length ($P_{\text{L}}$, Implicit Header (IH), Low Data Rate Optimization (DE), and Coding Rate (CR). The IH(IH) value is 0 if the header is enabled and 1 if it is not. Using an implicit header reduces the packet size by using predefined CR and Cyclic Redundancy Check (CRC) configurations; otherwise, these values are included in the frame header. If the low data rate optimization is enabled, it also affects the transmission time and DE value is set to 1.

In other words, the payload size is influenced by several factors:

- Packet Length (PL): The overall length of the data packet.

- Implicit Header (IH): If the header is included, IH is 0. If not, IH is 1. Using an implicit header makes the packet smaller by using predefined settings for the Coding Rate (CR) and Cyclic Redundancy Check Code (CRCC). If the header is not implicit, these settings are included in the packet.

- Low Data Rate Optimization (DE): If this feature is turned on, it also affects how long the transmission takes.

$$n_{\text{payload}} = 8 + \max\left(\left\lceil \frac{8P_L - 4SF + 28 + 16CR_{CC} - 20IH}{(CR+4)} \right\rceil, 0\right) \tag{5}$$
$$T_{\text{payload}} = n_{\text{payload}} \times T_{\text{sym}}$$

where:

- SF: Spreading Factor with values from 7 to 12
- CRC: Cyclic Redundancy Check mode. enabled is CRC = 1, disabled is CRC = 0 (default for LoRaWAN CRC = 1)
- IH: Implicit Header H = 1 or Explicit Header H = 0 (default for LoRaWAN H = 0)
- DE: Adaptive Data Rate. enabled is DE = 1, disabled is DE = 0 CR: Coding Rate. CR = 1, 2, 3, 4. (by default for LoRaWAN, CR = 1)

**LoRaWAN MAC Layer**

LoRa and similar low-power low-rate technologies must be highly efficient because any additional overhead can result in increased energy consumption or latency regarding frame airtime. To address this challenge, LoRaWAN employs a relatively straightforward channel management strategy to ensure that end devices remain cost-effective. LoRaWAN utilizes pure Aloha alongside additional ACK mechanisms to streamline medium access control. The MAC layer protocol defines one mandatory and two optional classes to accommodate various potential use cases.

- Class A support is required for all LoRaWAN end-devices, which allows for the transmission of an uplink message at any time. Following the uplink transmission, a Class A device opens two short receive windows, RX1 and RX2, with specific delays known as RX1 Delay and RX2 Delay. If the network server doesn't respond during these windows, the next downlink message will be scheduled immediately after the subsequent uplink transmission. In Class A, end devices initiate the Up-Link (UL) message transmission and open two Down-Link (DL) receive windows, as depicted in figure 2.4, providing the server flexibility to respond in either window. This class operates in the most power-efficient mode.

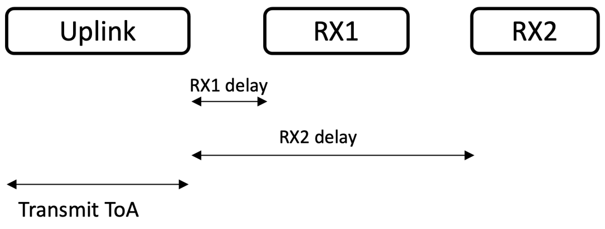


Figure 2.4: LoRaWAN Class A. Source [11].

- Class B devices build upon Class A capabilities by introducing periodic receive windows, known as ping slots, for downlink message reception. The network periodically broadcasts time-synchronized beacons through gateways, received by the end devices, serving as a timing reference to align their internal clocks with the network. This synchronization enables the network server to schedule downlink messages for specific devices or groups during the beacon period. Class B EDs, depicted in figure 2.5, are bi-directional and include scheduled receive slots in addition to the two receive windows from Class A. The gateway initiates time synchronization beacons to prompt the opening of receive windows at specific slots, allowing the server to confirm the ED's active listening status.

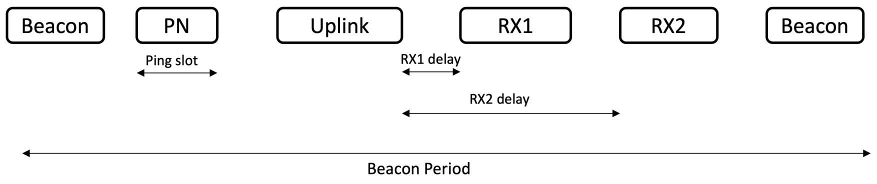


Figure 2.5: LoRaWAN Class B. Source [11].

- Class C devices extend Class A capabilities by keeping the receive windows open unless transmitting an uplink, as shown in the Figure 2.6. Therefore, Class C devices can receive downlink messages at almost any time, thus having very low latency for downlinks. These downlink messages can be used to activate certain functions of a device, such as reducing the brightness of a street light or turning on the cut-off valve of a water meter.

  Class C devices open two receive windows, RX1 and RX2, similar to Class A. However, the RX2 receive window remains open until the next uplink transmission. After the device sends an uplink, a short RX2 receive window opens, followed by a short RX1 receive window, and then the continuous RX2 receive window opens. This RX2 receive window remains open until the next uplink is scheduled. Uplinks are sent when there is no downlink in progress.

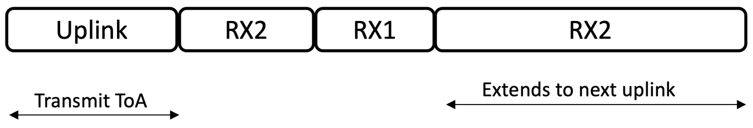


Figure 2.6: LoRaWAN Class C. Source [11].

EDs in Class C, illustrated in Figure 2.5, maintain almost continuous receive windows, consuming a higher amount of energy compared to other classes. Despite the increased energy consumption, the data transmission latency between the Network Server (NS) and the ED is minimal.

## 2.4 Internet Engineering Task Force and its role

The IETF is the premier organization responsible for the development and promotion of internet standards, particularly those that define the protocols used across the internet. One of the key areas of focus for the IETF is ensuring that networks can reduce the need for manual configuration and intervention. To this end, the IETF has established several working groups, with the ANIMA (Autonomic Networking Integrated Model and Approach) working group being central to the development of standards for Autonomic Networking.

The IETF's work in autonomic networking aims to create self-managing networks that can automatically configure, manage, and optimize themselves. This effort is crucial in modern networks, where the complexity and scale make manual management increasingly impractical. The protocols and frameworks developed by IETF, such as those within the ANIMA working group, are foundational to enabling these autonomic capabilities in network infrastructures.

## 2.5 Autonomic Networking

As mentioned in previous sections, IPv6, with its vastly expanded address space, presented a solution to the exhaustion of IPv4 addresses and the associated routing table complexities. By providing trillions of unique addresses, IPv6 offered the necessary foundation for the development of autonomic networking solutions. In response to the challenges posed by IPv4 exhaustion, network architects and managers began implementing IPv6 in their organizations. This transition not only addressed the scarcity of IP addresses but also allowed for the design of networks with optimized routing tables, thanks to the logical model of autonomic systems based on summary routes.

Autonomic refers to the ability of a system or entity to operate independently, without direct human intervention or control. In various contexts, autonomy implies self-governance, self-management, and self-regulation, but allowing high-level guidance by a central entity (intent) [12]. Autonomic systems are capable of making decisions, taking actions, and adapting to changing circumstances based on predefined rules, algorithms, or objectives.

Autonomic networking refers to the concept of self-managing and self-optimizing networks that can adapt and respond to changing conditions without human intervention. This approach leverages automation to streamline network operations, improve efficiency, enhance performance, and ensure reliability. Autonomic networking represents a paradigm shift from traditional network management approaches, empowering networks to become more agile, resilient, and intelligent. Here are some key aspects of autonomic networking:

- Self-Configuration: Autonomic networking enables networks to automatically configure and provision resources based on predefined policies and objectives. This includes tasks such as assigning IP addresses, configuring routing protocols, and setting up security policies. By automating these processes, networks can reduce deployment times, minimize human errors, and ensure consistency across network configurations.

- Self-Optimization: Autonomic networks continuously monitor their performance metrics and traffic patterns to identify optimization opportunities. Using AI and ML algorithms, networks can analyze data in real-time, predict future trends, and make proactive adjustments to optimize resource utilization, bandwidth allocation, and quality of service (QoS). This dynamic optimization improves network efficiency and responsiveness while maximizing user experience.

- Self-Healing: Autonomic networking enables networks to detect and respond to faults and failures automatically. By leveraging intelligent algorithms and automated remediation actions, networks can quickly identify and isolate issues, reroute traffic, and implement fail over mechanisms to maintain service continuity and minimize downtime. This self-healing capability enhances network reliability and resilience, ensuring high availability and fault tolerance.

### 2.5.1 Architecture

Autonomic networking architecture encompasses a hierarchical design model aimed at enabling self-management and self-configuration capabilities within network elements. This architecture comprises three interconnected layers.

- the Autonomic Network Infrastructure (ANI)

- the Autonomic Control Plane (ACP)

- the Autonomic Service Agent (ASA).

At the core of this architecture is the Autonomic Network Infrastructure (ANI), which acts as the backbone for the entire autonomic network. ANI enables a shared infrastructure among a group of autonomic nodes, allowing them to exchange services and information at the most fundamental level. Carpenter et al. [1] introduce the concept of autonomic nodes as Plug-and-Play devices within autonomic networks, highlighting the importance of self-securing as a key characteristic. Each autonomic node operates independently, requiring it to establish secure communication and initialize its ACP. This shared infrastructure not only facilitates communication between nodes but also provides essential data that supports the local functions of each node, as illustrated in Figure 2.7.

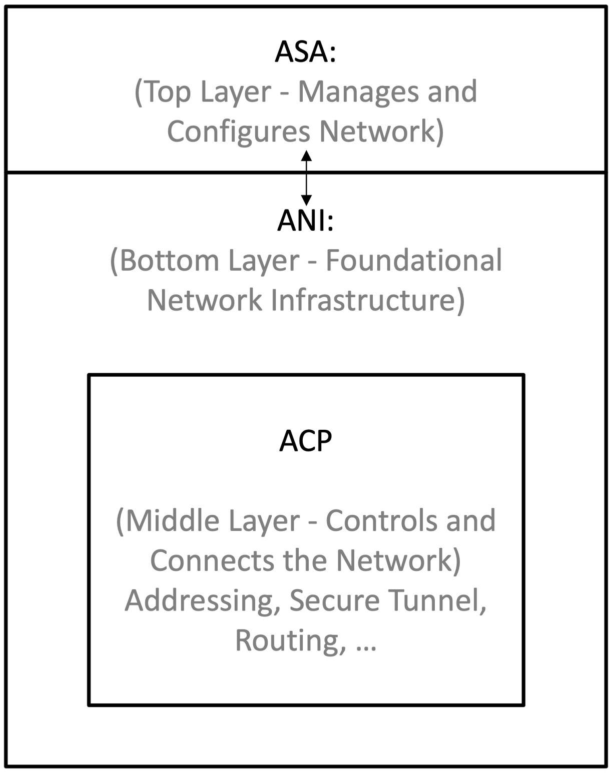

ASA:
(Top Layer - Manages and Configures Network)
ANI:
(Bottom Layer - Foundational Network Infrastructure)
ACP
(Middle Layer - Controls and Connects the Network)
Addressing, Secure Tunnel, Routing, …

Figure 2.7: Autonomic Node Architecture

Sitting atop the ANI is the ACP, functioning as the controller entity overseeing the nodes' operations. The ACP acts as an intermediary between the shared infrastructure and the higher layers, facilitating access to ANI services while providing critical information to the upper layers of the architecture.

The highest layer comprises the ASA, encompassing atomic entities and autonomic functions that enable autonomic nodes to leverage the services provided by the lower layers effectively. ASA integrates with the ACP to access shared infrastructure services and utilizes them to execute various autonomic functions, thereby empowering nodes to automatically manage and configure themselves.

### 2.5.2 Generic Autonomic Signaling Protocol (GRASP)

GRASP is a communication protocol defined by IETF [13], enabling autonomic nodes and ASAs to discover peers, synchronize states, and negotiate parameters dynamically. It supports decentralized, self-managing network operations, allowing nodes to coordinate actions such as resource management and network configuration without manual intervention.

### 2.5.3 Enrollment over Secure Transport (EST)

EST is a protocol defined by the IETF in the RFC7030 [14] to facilitate secure certificate enrollment for clients using the Public Key Infrastructure (PKI). EST operates by exchanging Certificate Management over CMS (CMC) messages over a secure transport layer, specifically leveraging TLS (Transport Layer Security) and HTTP to provide a secure, authenticated, and authorized channel for certificate management. The protocol enables clients to acquire certificates from a Certification Authority (CA) and manage key pairs, including both client-generated and CA-generated key pairs. EST is designed to be simple yet functional, ensuring that PKI clients can obtain necessary certificates and associated CA certificates securely, supporting a range of operations including initial enrollment, certificate re issuance, and distribution of CA certificates.

### 2.5.4 BRSKI

BRSKI (Bootstrapping Remote Secure Key Infrastructures) is an IETF-defined protocol that plays a pivotal role in the secure onboarding of new devices (as nodes) into an Autonomic Network. BRSKI can be used in conjunction with IPv6 protocols to automate the provisioning of IPv6 addresses and other network parameters in the end nodes.

BRSKI operates by leveraging existing security infrastructures, such as PKI and digital certificates, to authenticate devices. Once a device is authenticated, BRSKI provisions it with the required configuration to join the ACP securely. This process is vital for maintaining the integrity and security of the Autonomic Network.

The ACP, relies on BRSKI protocol for configuration. Multiple autonomic nodes play distinct roles in the bootstrapping process. BRSKI facilitates mutual authentication through the exchange of X.509 certificates via an external entity, assisting in this authentication. The X.509 certificates serve as a public key certificate format in secure transport protocols.

The following includes definitions for components and nodes used in BRSKI.

- **Pledge:**
  Pledge is a critical component in establishing secure connections for Internet of Things (IoT) devices. It facilitates the enrollment of devices into a network securely, ensuring that only authorized devices gain access. The pledge is responsible for initiating the bootstrapping process, where it generates a public-private key pair and communicates with a registrar to obtain necessary credentials for network access. This process is crucial for IoT deployments, especially in scenarios where manual intervention for device setup is impractical or insecure. BRSKI pledge streamlines the onboarding process, enhancing the overall security posture of IoT networks.

- **Join Proxy:**
  To address security concerns, the registrar cannot openly disclose its presence, as it might pose a security risk. Richardson et al. [1] introduce the join proxy, a low-security risk intermediary role connecting the pledge to the authenticated registrar. The pledge, aware that it must pass through the join proxy to reach the registrar, seeks a join proxy among its neighbors using DULL GRASP. Upon finding the join proxy, the pledge dispatches a voucher request message directed to the registrar but relayed through the join proxy.

- **Registrar:**
  Registrar component is to securely onboarding IoT devices onto a network. It acts as a central authority responsible for verifying the identity of devices seeking access and provisioning them with the necessary credentials to join the network securely. The registrar facilitates the enrollment process by authenticating the identity of devices through mechanisms such as certificates or pre-shared keys. Once authenticated, the registrar provides the devices with the necessary information, including cryptographic material and network configuration parameters, enabling them to establish secure

connections with other network components. By overseeing the bootstrapping process, the BRSKI registrar ensures that only authorized devices gain access to the network, thereby enhancing overall network security and integrity.

- **Manufacturer Authorized Signing Authority (MASA):**

  The MASA operates independently of the domain but holds a critical function in authenticating the pledge. Serving as a representative of the pledge device's manufacturer, MASA maintains a registry of device owners, both current and past, for ownership verification. The manufacturer incorporates a certificate within the pledge device, allowing MASA to conduct device authentication. Notably, the pledge does not establish direct communication with MASA; instead, the registrar takes on the responsibility of requesting MASA to authenticate both the device and its owner. In this process, MASA also authenticates the registrar to the pledge, facilitating mutual authentication between the pledge and the registrar. Hence, MASA plays a pivotal role in ensuring the mutual authentication of the pledge and the registrar.

- **Voucher:**
  In BRSKI, a voucher is a cryptographic message that securely communicates authorization and configuration details from the registrar to the pledge during enrollment. It contains essential network parameters signed by a trusted authority, enabling the pledge to securely configure itself and establish trust within the network.

BRSKI answers four main questions:

(1) The registrar verifies the identity of the pledge by asking, "Who is this device? What is its identity?"

(2) The registrar decides whether to authorize the pledge by questioning, "Is it mine? Do I want it? What are the chances it has been compromised?"

(3) The pledge verifies the identity of the registrar by asking, "What is this registrar's identity?"

(4) The pledge decides whether to authorize the registrar by considering, "Should I join this network?"

BRSKI outlines protocols and messages to address the aforementioned inquiries. It employs a TLS connection and a PKIX-shaped (X.509v3) certificate, such as an IEEE 802.1AR IDevID, to address points 1 and 2. Additionally, it introduces voucher as a new concept, obtained by the registrar from an external entity and passed to the pledge, to address points 3 and 4.

### 2.5.5 Management, Control, and Data Planes

In modern networking, three key planes - control, data, and management - work together seamlessly to ensure network efficiency. The planes are shown in Figure2.8. Control Plane manages routing protocols, deciding the best paths for data transmission across the network using protocols like OSPF and BGP. Data Plane, also called the forwarding plane, it physically forwards data packets based on routing information provided by the control plane, using protocols like Ethernet and IP. Management Plane handles administrative tasks like device configuration, performance monitoring, and security management using protocols such as SNMP and NETCONF, ensuring network stability and efficiency.

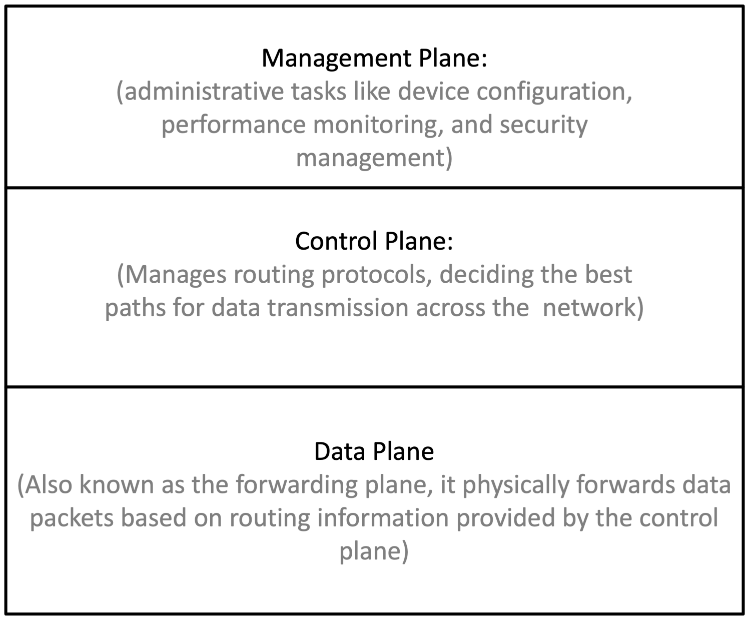

Management Plane:
(administrative tasks like device configuration, performance monitoring, and security management)
Control Plane:
(Manages routing protocols, deciding the best paths for data transmission across the network)
Data Plane
(Also known as the forwarding plane, it physically forwards data packets based on routing information provided by the control plane)

Figure 2.8: Management, Control, and Data Planes

## 2.6 IP Networking in IoT

Internet Protocol (IP) networking forms the backbone of the Internet of Things (IoT), enabling connectivity and functionality across a diverse array of devices. Within IoT networks, IP addressing and routing facilitate seamless communication and data exchange among devices, supported by wireless connectivity technologies like Wi-Fi and Bluetooth. Protocols such as MQTT and CoAP optimize data transmission over low-power and low-bandwidth networks often used in IoT applications, ensuring efficient communication.

### 2.6.1 IPv6

IPv6 plays a pivotal role in the Internet of Things (IoT) ecosystem for several reasons. Firstly, its expansive address space provides trillions of unique addresses, accommodating the escalating number of IoT devices, a stark contrast to the limited pool of IPv4 addresses. This abundance is crucial for accommodating the surging number of IoT devices, ensuring each device can have its distinct identifier without relying on complex network address translation mechanisms.
Additionally, IPv6's scalability is indispensable for the growing IoT landscape, as it ensures there are ample unique addresses to support the burgeoning number of connected devices. This scalability facilitates seamless integration of new IoT devices into existing networks without facing address exhaustion issues.
Furthermore, IPv6 offers autoconfiguration capabilities, enabling IoT devices to acquire IP addresses automatically, simplifying network deployment and management. Its efficiency in packet processing and routing, along with built-in support for IPsec, enhances network performance and security, making it ideal for resource-constrained IoT devices. Lastly, IPv6 adoption future-proofs IoT deployments, ensuring long-term compatibility and support for emerging technologies, thus averting the need for costly network upgrades.

### 2.6.2 IPv6 addresses

- **Global Unicast Address (GUA):**
  This is the equivalent of a public IP address in IPv6, globally unique and routable on the IPv6 internet.

Typically, these addresses start with a prefix of 2000::/3, which indicates that they are part of the global unicast address space. This address is used for communication across the internet, ensuring that a device can be reached from any other device on the global IPv6 network.

- **Link Local Address:**
  This address is used for communication between devices on the same local network segment or link. It is not routable beyond the local link, meaning it cannot be used to communicate across different networks. These addresses start with the prefix FE80::/10 and are automatically generated by devices when they connect to an IPv6-enabled network. The primarily usage of this address is for neighbor discovery, link-layer addressing, and other local network management tasks.

- **Unique Local Address (ULA):**
  This address is similar to private IP addresses in IPv4 (like 192.168.x.x), meant for use within a specific organization or site. It is unique within a local environment but not globally routable on the internet. ULAs start with the prefix FC00::/7, with the most common implementation using FD00::/8 for locally assigned addresses. This address is used for internal network communication, such as within a corporate network or between devices in a home network, without exposing the addresses to the global internet.

### 2.6.3 IPv6 Autonomic Behavior

IPv6 autonomic behavior refers to the capability of IPv6 networks to self-manage and self-configure with minimal human intervention. IPv6 autonomic behavior extends into various mechanisms and protocols that contribute to the seamless, self-managing nature of modern networks. Key among these are address auto configuration and Neighbor Discovery (ND), which are explained in the next chapter. These technologies work together to create networks that are not only more efficient but also capable of adapting to changing conditions without manual intervention.

One of the autonomic behaviour in IPv6 is the dynamic SCHC updates. This term is essential for maintaining efficient communication in networks with constrained devices, such as IoT environments. As these networks evolve—whether due to changes in device roles, application requirements, or network configurations—the predefined SCHC contexts, which dictate how headers are compressed, may no longer be optimal. Dynamic SCHC context updates allow these compression rules to be adjusted in real-time, ensuring that the network continues to operate efficiently without manual intervention.

### 2.6.4 IP supporting LPWAN

Supporting Internet Protocol (IP) communication over Low-Power Wide-Area Networks (LPWANs) faces challenges due to their small packet sizes, constrained bandwidth, and limited energy. Efforts are underway to enable seamless IP connectivity in LPWANs by adapting protocols like IPv6 and using header compression techniques. A key advancement is the integration of 6LoWPAN, which addresses LPWAN challenges with mechanisms like header compression, address autoconfiguration, fragmentation, and neighbor discovery. Header compression reduces IPv6 and UDP overhead, making data transmission efficient, while fragmentation enables larger packet transmission. 6LoWPAN Neighbor Discovery optimizes IPv6 Neighbor Discovery for LPWANs, facilitating efficient IP communication critical for IoT deployment.

6LoWPAN aimed to adapt IPv6 to the constraints of low-power wireless networks by defining header compression techniques, fragmentation mechanisms, and other optimizations to reduce overhead and conserve bandwidth. However, over time, it became apparent that 6LoWPAN had limitations and wasn't fully suitable for all LPWAN technologies and use cases.

As a result, the IETF established the Static Context Header Compression (SCHC) working group to develop a more flexible and efficient solution for IP-based communication over LPWANs. SCHC introduces a standardized framework for compressing IPv6/UDP/CoAP headers, allowing efficient transmission of IP packets over LPWANs with constrained resources.

## 2.7 Static Context Header Compression and Fragmentation (SCHC)

This section introduces the SCHC adaptation layer, positioned between IPv6 and an underlying LPWAN technology, which comprises two crucial sublayers: header compression and fragmentation. The subsequent subsections delineate the key design principles and features inherent in these sublayers. Developed by IETF, the SCHC specification facilitates header compression for IPv6, UDP, and CoAP protocols within LPWAN technologies . This compression and fragmentation capability enables communication over the Internet directly to the assigned IP address of an end device [10]. The specific LPWAN technology addressed in this context is LoRaWAN, and the operational guidelines for SCHC over LoRaWAN are defined in RFC 9011 [15].

SCHC (Static Context Header Compression) emerges as a more effective solution for addressing the stringent constraints of Low Power Wide Area Network (LPWAN) technologies compared to 6LoWPAN. While 6LoWPAN provides some compression capabilities for IPv6 and UDP headers, SCHC offers a higher compression ratio, potentially reducing a 48-byte uncompressed header to a single byte. This enhanced compression efficiency is vital for optimizing data transmission over LPWANs with extremely limited bandwidth and packet sizes. Moreover, SCHC is tailored explicitly for LPWAN technologies, ensuring its compression and fragmentation functionalities are optimized to effectively address these constraints, leading to improved resource utilization and network performance. Additionally, SCHC supports CoAP header compression, a feature lacking in 6LoWPAN, further enhancing its potential for efficiency and performance gains in 6LoWPAN environments.

### 2.7.1 SCHC Compression

The efficient transmission of IP-based protocols over LPWANs is hindered by the significant overhead introduced by typical packet header sizes, which are notably larger than the extremely limited LPWAN frame payload sizes. While various header compression mechanisms have been developed for efficient packet transmission over different technologies, the Robust Header Compression (ROHC) and 6LoWPAN header compression mechanisms, designed for IP-based packet headers, are not suitable for LPWANs. In response to these challenges, the Static Context Header Compression (SCHC) has been purposefully designed for LPWANs, specifically applicable to protocols like IPv6, UDP, and CoAP. [16].

SCHC relies on a static context shared between the compressor and the decompressor, leveraging a priori knowledge of the traffic to be compressed. This approach eliminates the need for context resynchronization mechanisms and receiver feedback, enabling ultra-lightweight header compression. In SCHC, a context is defined as a set of rules, each associated with a Rule identifier (Rule ID). Each rule describes how to compress specific packet header fields, and a rule may be used for compressing one or more protocol headers. When sending a packet, the SCHC compressor selects the rule that best matches the header format and values of the packet, replacing the original packet header with the corresponding Rule ID. In cases where a Rule ID cannot unambiguously represent a complete packet header, a compression header residue is generated, constituting the compressed header. Upon receiving a compressed packet, the decompressor reconstructs the original packet header based on the received compressed header and the stored context. The compact size of the Rule ID allows for efficient encoding of a large number of rules, ensuring flexibility and scalability in LPWAN environments [17].

### 2.7.2 SCHC Stack Flow

The requirement in IPv6 for any underlying layer to support packets of at least 1280 bytes, intended to achieve high performance over a presumed resource-rich Internet, poses challenges in LPWAN networking. LPWANs, designed for infrequent message exchanges with short-sized payloads, often have extremely short maximum frame payload sizes, some even as low as 10 bytes. Despite the highly efficient SCHC header compression, many IPv6 packets may not fit into a single LPWAN frame, and neither LoRaWAN nor Sigfox supports fragmentation and reassembly. To address this, SCHC introduces fragmentation at its adaptation layer, positioned below the header compression sublayer [16].

In the context of LoRaWAN, the SCHC compression sublayer processes IPv6 packets, resulting in SCHC Packets. If the SCHC Packet is within the LPWAN protocol's Maximum Transmission Unit (MTU), it is transmitted without fragmentation; otherwise, it is delivered to the fragmentation sublayer. The transmission process through SCHC sublayers is illustrated in Figure 2.9, with the fragmentation sublayer and its modes explained below as a theoretical basis for understanding efficiency in this sublayer [10].

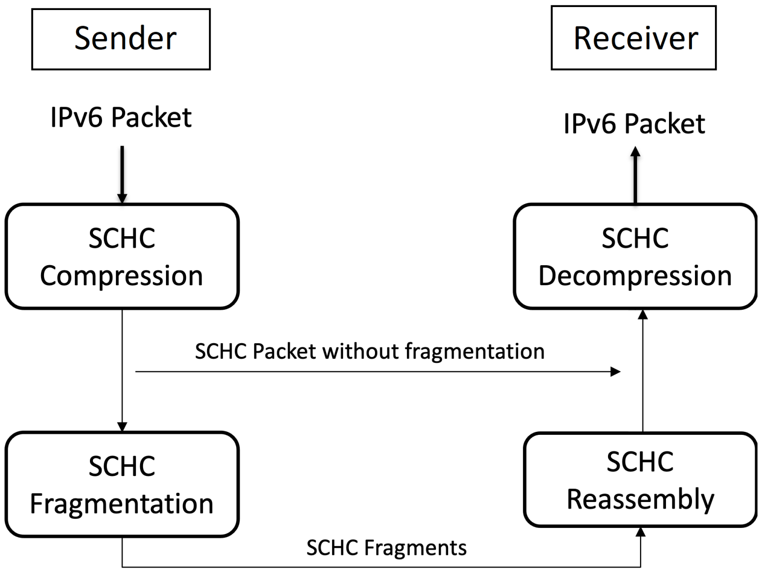


Figure 2.9: SCHC Stack Flow. Adapted from [2]

### 2.7.3 SCHC Fragmentation

A SCHC Fragment consists of a SCHC Fragment Header and payload, where the payload carries one tile. For LoRaWAN, the SCHC Fragment Header includes the window number (W) and FCN for the tile (Figure 2.10).

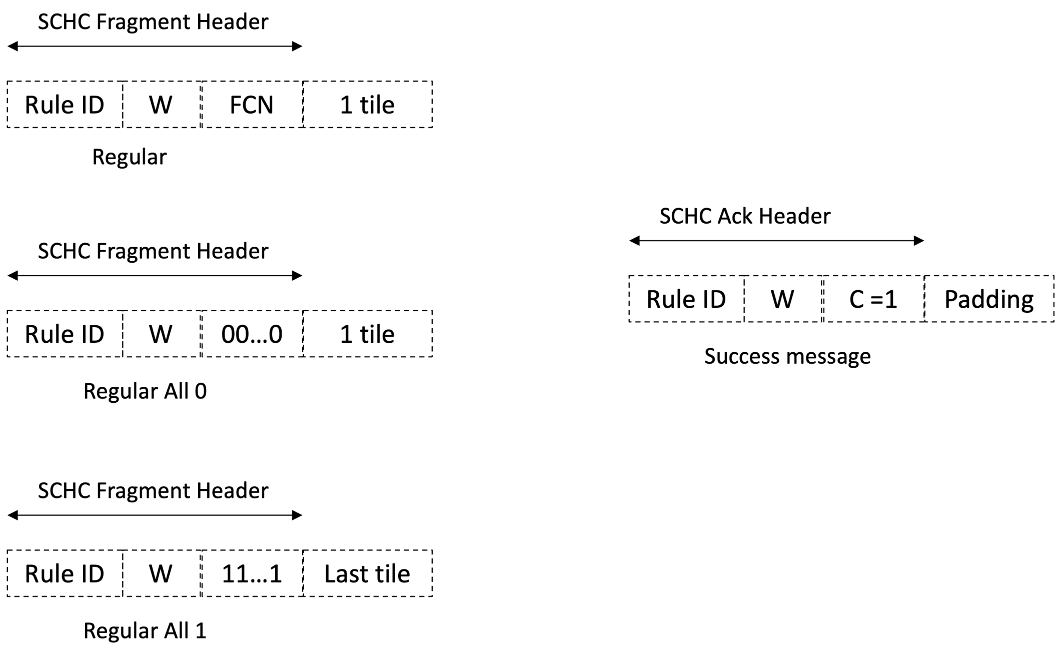


Figure 2.10: SCHC Fragment. Adapted from [2]

Preceding these is the RuleID, indicating the SCHC Fragment Header size. Two types of SCHC Fragments exist: Regular (excluding the last tile of each window) and All-0 SCHC Fragments (carrying the last tile of each window except the last one). The All-1 SCHC Fragment denotes the end of the SCHC Packet. The SCHC ACK message includes a SCHC ACK Header and a bitmap. The header contains the RuleID, window number (W), and integrity check bit (C), with C = 1 indicating a successful transfer. When C = 1, the bitmap is omitted for optimization, with padding added if needed for alignment with the underlying L2's minimum data unit.

The figure 2.11 illustrates the flow of SCHC ACK mode packets transfer. In the event that a receiver receives an All-0 SCHC fragment, it only sends a SCHC acknowledgment (ACK) as shown in Figure 2.11.

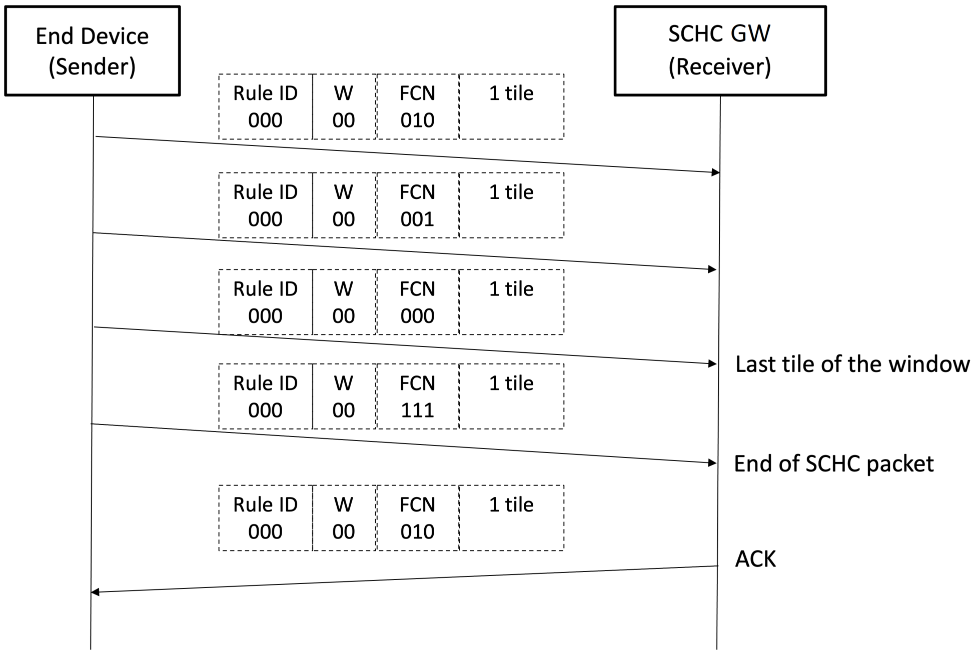


Figure 2.11: SCHC message flow. Source [15]

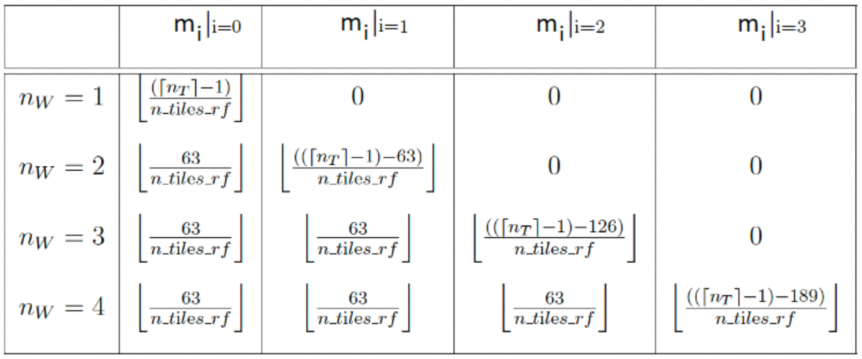
To calculate the window and fragment time we used below formula adapted from Muñoz-Lara in [2]:

$$W_i = m_i * FRF + PRF + AR \tag{3}$$

Where:

- $W_{\mathrm{i}}$: number of the windows
- $m_{\mathrm{i}}$: number of full fragments
- FRF: time of the full size fragments
- PRF: time of the Partial size fragment
- AR: Ack request and response time

In [2], the authors introduced Table 2.12 that shows how $m_i$ is calculated for each window and fragment:



| | $\mathrm{m_i}\|_{i=0}$ | $\mathrm{m_i}\|_{i=1}$ | $\mathrm{m_i}\|_{i=2}$ | $\mathrm{m_i}\|_{i=3}$ |
|---|---|---|---|---|
| $n_W = 1$ | $\left\lfloor \frac{(\lceil n_T \rceil - 1)}{n\_tiles\_rf} \right\rfloor$ | $0$ | $0$ | $0$ |
| $n_W = 2$ | $\left\lfloor \frac{63}{n\_tiles\_rf} \right\rfloor$ | $\left\lfloor \frac{((\lceil n_T \rceil - 1) - 63)}{n\_tiles\_rf} \right\rfloor$ | $0$ | $0$ |
| $n_W = 3$ | $\left\lfloor \frac{63}{n\_tiles\_rf} \right\rfloor$ | $\left\lfloor \frac{63}{n\_tiles\_rf} \right\rfloor$ | $\left\lfloor \frac{((\lceil n_T \rceil - 1) - 126)}{n\_tiles\_rf} \right\rfloor$ | $0$ |
| $n_W = 4$ | $\left\lfloor \frac{63}{n\_tiles\_rf} \right\rfloor$ | $\left\lfloor \frac{63}{n\_tiles\_rf} \right\rfloor$ | $\left\lfloor \frac{63}{n\_tiles\_rf} \right\rfloor$ | $\left\lfloor \frac{((\lceil n_T \rceil - 1) - 189)}{n\_tiles\_rf} \right\rfloor$ |

Figure 2.12: Window time and fragment time calculation. Adapted from [2]

### 2.7.4 SCHC Context

The SCHC methodology capitalizes on the predictable nature of traffic in typical LPWAN applications, allowing both communication endpoints to leverage a pre-shared, static context containing header information. This context comprises a set of rules identified by a fixed-size Rule-ID, enhancing flexibility and performance by dynamically adjusting the compression configuration based on real-time traffic. In scenarios with low traffic variability, where changes in header fields are minimal, predefined header configurations (Rule-ID) are stored at both the source and destination endpoints. Consequently, the identifier of the rule closest to the uncompressed header is transmitted, accompanied by a potential residue field for information not retrievable from the Rule-ID alone [18]. In cases where the residue is empty, the most concise message length is achievable. Each rule encompasses various fields derived from the original header parameter, including Field ID (FID), Field Length (FL), Field Position (FP), Direction Indicator (DI), Target Value (TV), Matching Operator (MO), and Compression Decompression Action (CDA) [19]. This approach optimizes header compression and decompression processes, as depicted in the simplified graphical representation in Figure 2.13.

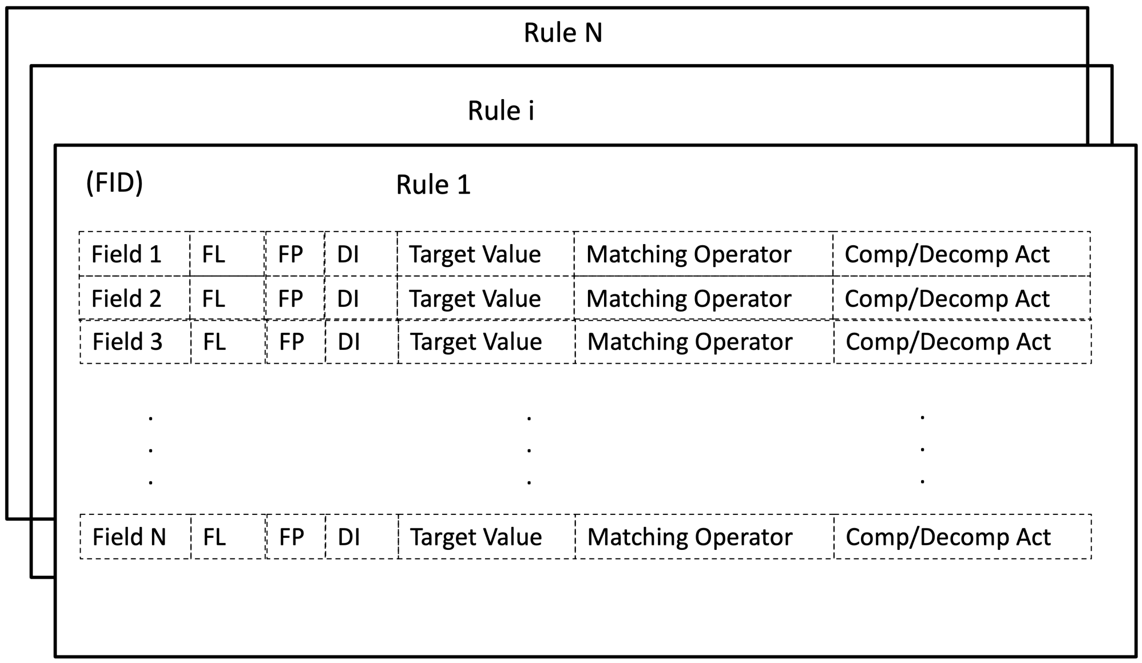


Figure 2.13: SCHC Context. Source [19].

Let's discuss each component in detail:

- Field ID (FID): FID identifies a specific field within the header that is targeted for compression or decompression.Each field in the header is assigned a unique FID to facilitate compression and decompression operations.
- Field Length (FL):FL specifies the length of the compressed representation of the field. It indicates the number of bits allocated for representing the field's value in the compressed form.
- Field Position (FP):FP determines the position of the compressed field within the compressed header.

It indicates the offset or position where the compressed representation of the field starts in the compressed header.

- Direction Indicator (DI):DI indicates the direction of the packet flow (uplink or downlink) for which the compression or decompression rules apply. It ensures that compression and decompression operations are applied correctly based on the packet's direction.
- Target Value (TV):TV represents the desired value of the field to be matched during decompression. It serves as a reference value against which the received compressed field is compared to reconstruct the original field value during decompression.
- Matching Operator (MO):MO specifies the operation used to compare the received compressed field with the target value during decompression. Common matching operators include exact match, prefix match, suffix match, and range match, allowing for flexible field reconstruction.
- Compression Decompression Action (CDA):CDA defines the actions to be performed during compression and decompression operations. Compression actions include truncation, repetition, and omitting the field, while decompression actions involve reconstructing the original field value based on the compressed representation and target value.

By leveraging these components and mechanisms, SCHC enables efficient compression and decompression of IPv6/UDP/CoAP headers in LPWAN environments, reducing overhead and optimizing bandwidth utilization without compromising interoperability or reliability. SCHC provides a standardized and interoperable solution for header compression in LPWAN deployments, facilitating seamless integration with existing IP-based protocols and applications.

The table 2.1 shows the maximum payload sizes allowed in each SF and their corresponding number of tiles in each fragment.

Table 2.1: Maximum size for a full fragment payload. Adapted from [2]

| **SF** | **Maximum payload size** | **Tiles for each regular fragment** |
|---|---|---|
| 12 | 51 | 5 |
| 11 | 51 | 5 |
| 10 | 115 | 5 |
| 9 | 222 | 11 |
| 8 | 222 | 22 |
| 7 | 222 | 22 |

# Chapter 3

# Onboarding

In the previous chapter, we explained the terms related to our research. Now that LoRaWAN, IPv6, and BRSKI are understood concepts, we will demonstrate how these technologies are applied in onboarding process. Configuring and setting up new devices is a crucial first step before any data can be transferred. This chapter will delve into several key areas: the process of onboarding devices in a LoRaWAN network, the mechanisms of onboarding with autonomic networking perspective, and the configuration of device addresses within an IPv6 network. Additionally, we will evaluate the security metrics considered in each scenario to ensure robust and secure IoT deployments.

## 3.1 LoRaWAN

This section describes the onboarding process of LoRa devices to the LoRaWAN network, called device activation. Following this, an overview of the security metrics and mechanisms designed to protect against threats during LoRaWAN device activation will be provided.

### 3.1.1 Over The Air Activation Process (OTAA)

LoRaWAN supports two primary end-device activation methods: 1. Over-the-Air Activation (OTAA): This method involves devices being provisioned and authenticated over the air before joining the network. 2. Activation by Personalization (ABP): With this method, devices are pre-configured with security credentials and join the network by personalization.

Given its enhanced security and recommendation for end-device activation, the OTAA method is the primary focus of this thesis. Unlike alternative methods, OTAA eliminates the need for hardcoding device addresses and security keys into the device.

Here are the steps involved in Over The Air Activation (OTAA) within the LoRaWAN framework:

- Initialization: In OTAA, end-devices commence a join procedure by engaging in an authentication message exchange with the Network Server (NS) before establishing data communication.
- Join Procedure: The join procedure encompasses a join-request transmitted from the end-device to the NS and a subsequent join-accept transmitted from the NS to the end-device.

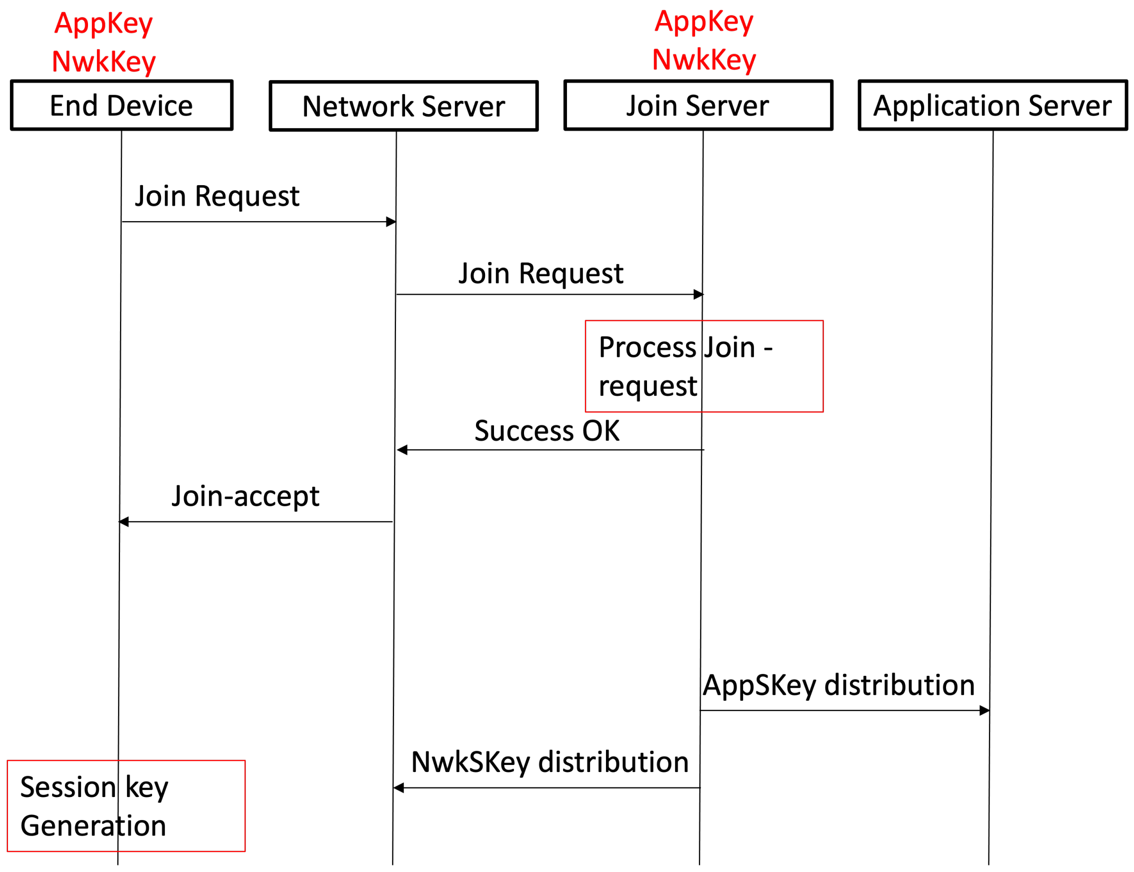


Figure 3.1: OTAA message flow in LoRaWAN. Adapted from [20]

- Define Device: Before initiating the join procedure, an end-device is defined by specific information, including a globally unique end-device identifier (DevEUI), the join server identifier (JoinEUI), and an advanced encryption standard key (AppKey).
- Recovery Procedure: If an end-device loses its network session information, it triggers a new join procedure to regain network access.

### 3.1.2 Security mechanisms

LoRaWAN is designed to offer secure communication for IoT devices over long distances while maintaining low power consumption. The security framework encompasses both front-end and back-end security mechanism to ensure data integrity, confidentiality, and authentication. Here is an overview of the key security metrics and mechanism which can be shown in the figure 3.2 as well:

#### Frontend security

The Key Management and Exchange during OTAA uses AES Encryption for Network and Application Layers. Two keys are provided in the process to be shared; Network Session Key (NwkSKey) between the end device and the network server, and Application Session Key (AppSKey) between end device and application server

- Network Session Key (NwkSKey): A Network session key employed for encrypting communication between end-devices and the network server. It provides data integrity and authentication between the end device and the network server. It uses AES-CTR to ensure message confidentiality.
- Application Session Key (AppSKey): An Application session key (AES-128 key) utilized to protect application-specific data. Ensures data confidentiality between the end device and the application server. It uses AES-CMAC for message integrity and authentication.

Note that, the initial Join Request message sent from a node is unencrypted. However, the response message from the Network Server (Join Accept message) is encrypted with the AppKey. NwkSKey and AppSKey are calculated as follows:

$$\text{NwkSKey} = \text{aes128\_encrypt}(\text{AppKey}, 0x01|\text{AppNonce}|\text{NetID}|\text{DevNonce}|\text{pad16})$$

$$\text{AppSKey} = \text{aes128\_encrypt}(\text{AppKey}, 0x02|\text{AppNonce}|\text{NetID}|\text{DevNonce}|\text{pad16})$$

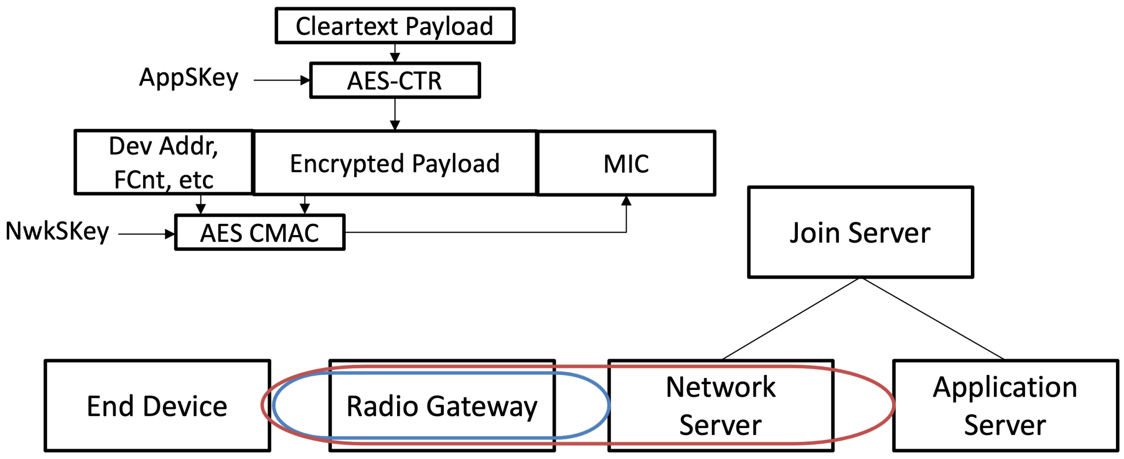


Figure 3.2: OTAA Key encryption. Source [21].

### Backend security

There are a few optional ways to provide the backend security before the OTAA starts, so the infrastructre is safe to start the key sharing between devices. TLS is one of the secure mechanisms used to secure communication between network servers and application servers, ensuring data is encrypted and safe from eavesdropping during transit. Firewalls and Machine Learning mechanisms are also useful options to protect the backend infrastructure from unauthorized access and to detect any malicious activities.

### Prevention against attacks

The OTAA Key encryption provides security mechanism which are all based on AES-128. They prevent LoRaWAN communication from the attacks listed below:

- Unauthorized Access: Unauthorized Access provides mutual end-point authentication to prevent unauthorized devices or users from accessing the network. The AppKey and NwkKey are never sent over the air, ensuring they remain confidential and are known only to the device and the network server, not exposed to unauthorized entities. This ensures that only devices with the correct keys can decrypt subsequent communication and join the network.

- Spoofing: Spoofing involves an attacker impersonating a legitimate device or entity to gain unauthorized access or information. In a network context, this includes sending messages that appear to come from a trusted source. A replay attack, a type of spoofing, occurs when the attacker intercepts and retransmits valid data to deceive the receiver.
  In protocols like LoRaWAN, replay protection is achieved through the use of unique nonces (e.g., DevNonce in the join procedure). Each message includes a nonce that must be unique for every transaction. This ensures that previously captured messages cannot be reused. In LoRaWAN, DevNonce is incremented with every Join-request to prevent replay attacks. A Join-request with a reused DevNonce will be rejected by the Join Server, preventing attempts to reuse old messages to gain unauthorized access.

- Modification: Each Join-request and Join-accept message includes a MIC, calculated using a cryptographic function and the NwkKey. This MIC ensures the message has not been altered in transit. The receiver recalculates the MIC and compares it to the received MIC to verify integrity.

- Eavesdropping: Eavesdropping is the unauthorized interception and listening to private communication or data transmission between parties to gain information without their knowledge or consent. OTAA prevents eavesdropping by encrypting the Join-accept message with the NwkKey and ensures ongoing encryption by deriving unique session keys (AppSKey and NwkSKey) for secure data transmission.

## 3.2 IPv6

IPv6 addresses can be configured manually, via DHCPv6, or using Stateless Address Auto-configuration (SLAAC). SLAAC, a required functionality of IPv6, allows devices to self-configure unique addresses automatically, without the need for a central server. SLAAC ensures unique and globally routable addresses. In this thesis, we will describe SLAAC, since it allows devices to automatically configure their addresses without manual intervention, promoting a more self-configuring network.

### 3.2.1 Neighbor Discovery Protocol

Neighbor Discovery is explained in this section because it is a fundamental protocol that underpins many of the key processes involved in IPv6 SLAAC. SLAAC relies on ND to facilitate essential functions such as generating link-local addresses, detecting duplicate addresses, and obtaining global network prefixes from routers.
The ND protocol is a key component of IPv6 that enables devices on the same local network to discover each other, determine each other's link-layer addresses, and manage reachability information. NDP replaces several IPv4 protocols like ARP and ICMP Router Discovery, providing a more efficient and flexible mechanism for network communication. It plays a crucial role in the auto-configuration process, as it allows devices to discover routers, obtain network prefixes, and ensure that their addresses are unique on the local network segment. NDP includes important mechanisms such as Router Solicitation, Router Advertisement,

Neighbor Solicitation, Neighbor Advertisement, and Redirect messages, all of which contribute to the seamless operation of an IPv6 network.

### 3.2.2 SLAAC

In this mechanism, the End Device automatically configures its own IPv6 link-local address to connect to an IPv6-enabled network. This local address allows the device to communicate at Layer 3 with other IPv6 devices within the same local network segment. The most common method for creating a link-local address is to derive the EUI-64 interface identifier from the device's MAC address (or DevEUI address in LoRa), and combine it with the link-local prefix FE80::/64. These are the steps included in SLAAC:

- First, the node Configures Itself with a Link-local Address: When a device connects to an IPv6 network, it first auto-configures a link-local address using the prefix FE80::/64 and an identifier derived from its EUI-64 interface identifier. LoRa device uses its 64 bit globaly unique DEvEUI as its EUI-64 interface identifier. This is an example to show how to create an IPv6 Link Local Address:

  1. DevEUI Address: 70-07-12-34-56-78

  2. Split the DEvEUI Address into Two Halves:
     70-07-12 and 34-56-78

  3. Insert 0xFFFE in the Middle:
     70-07-12-FF-FE-34-56-78

  4. Convert the First Byte (70) to Binary: 70 in binary =
     0111 0000

  5. Invert the U/L (Universal/Local) Bit (7th bit from the left):
     0111 0000 (original)
     0111 0010 (inverted 7th bit)

  6. Convert Back to Hexadecimal: 0111 0010 in binary =
     72 in hex

  7. Combine All Parts to Form the EUI-64 Interface Identifier:
     EUI-64 Identifier: 72-07-12-FF-FE-34-56-78

  Finally, the Link-local Address is formed by Adding the FE80::/64 Prefix as provided in the Figure 3.3.

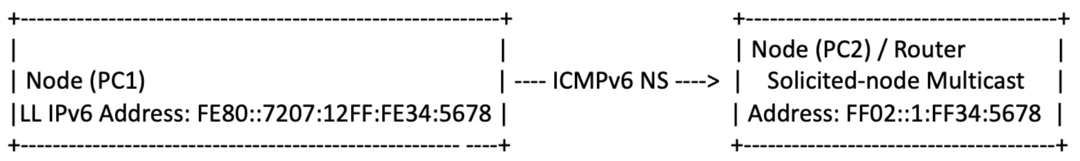


Figure 3.4: NS message to multicast address

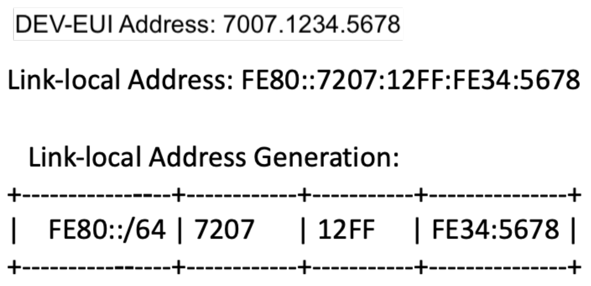

DEV-EUI Address: 7007.1234.5678
Link-local Address: FE80::7207:12FF:FE34:5678
Link-local Address Generation:
FE80::/64
7207
12FF
FE34:5678

Figure 3.3: Link local address generation

- Duplicate Address Detection (DAD): DAD is a critical part of the Neighbor Discovery protocol in IPv6. It ensures that any IP address assigned to a device, whether link-local or global, is unique on the local network segment. The device ensures its link-local address is unique by performing DAD, where it sends an ICMPv6 Neighbor Solicitation message to a solicited-node multicast address. If no response is received, the address is unique. Here are the example steps in detail:

  1. In thus step, the last 6 hex digits of the IPv6 address is appended to the multicast group prefix FF02::1:FF00:0/104 to form the solicited-node multicast address, which will be:
     FF02::1:FF34:5678.

  The node sends a Neighbor Solicitation (NS) message to the solicited-node multicast address as shown in Figure 3.4:

  2. If not subscribed to multicast group FF02::1:FF34:5678, no Neighbor Advertisement (NA) response will be received from the multicast group PC or routers as shown in Figure 3.5.

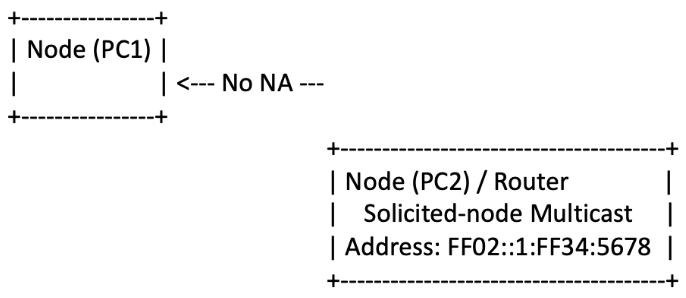


Figure 3.5: NA response from multicast group

3. Finally, if not receiving any NA messages, the address is considered unique and is assigned to the node:

    LL IPv6 Address: FE80::7207:12FF:FE34:5678

The next steps are for configuring the global unicast address.

- Router Solicitation Message: The device then sends a Router Solicitation message to discover network routers and obtain the global IPv6 prefix. The routers respond with Router Advertisement messages containing the necessary prefix information.
- Node Configures Its Global Unicast Address: The device combines the received prefix with its EUI-64 interface identifier to create a global unicast address. It also sets its default gateway to the router's link-local address. This is an example for global unicast address according to the example in the first step:

  Combining Prefix with Interface Identifier:
      Prefix: 2001:1234:A:B::/64
      EUI-64 Interface ID: 7207:12FF:FE34:5678
      Global Unicast Address: 2001:1234:A:B:7207:12FF:FE34:5678/64
- Duplicate Address Detection for Global Address: The device performs DAD again for its global unicast address by sending a Neighbor Solicitation message. If no reply is received, the address is confirmed as unique and can be used for communication.

Overall, in IPv6 auto-configuration, devices configure a link-local address using an EUI-64 identifier derived from the MAC address, and ensure the uniqueness of this address through Duplicate Address Detection (DAD). For global communication, devices request network prefixes from routers via Router Solicitation messages and configure their global unicast addresses by combining the received prefix with their EUI-64 identifier, followed by another DAD to verify uniqueness. This self-configuring approach simplifies network management and promotes seamless connectivity within IPv6 networks.

## 3.3 BRSKI

The Bootstrapping Remote Secure Key Infrastructure (BRSKI) [22] protocol plays a crucial role in securely onboarding new devices into a network. It automates the initial security configuration of devices, ensuring they can communicate securely and be managed effectively. This process involves a series of structured messages exchanged between the device being onboarded (referred to as the bootstrapped), the local network infrastructure, and the manufacturer's trusted services. The following shows an overview of the messages and their roles in the onboarding process. Finally, an overview of the security mechanism and mechanisms designed to protect against threats during the BRSKI bootstrapping process is provided.

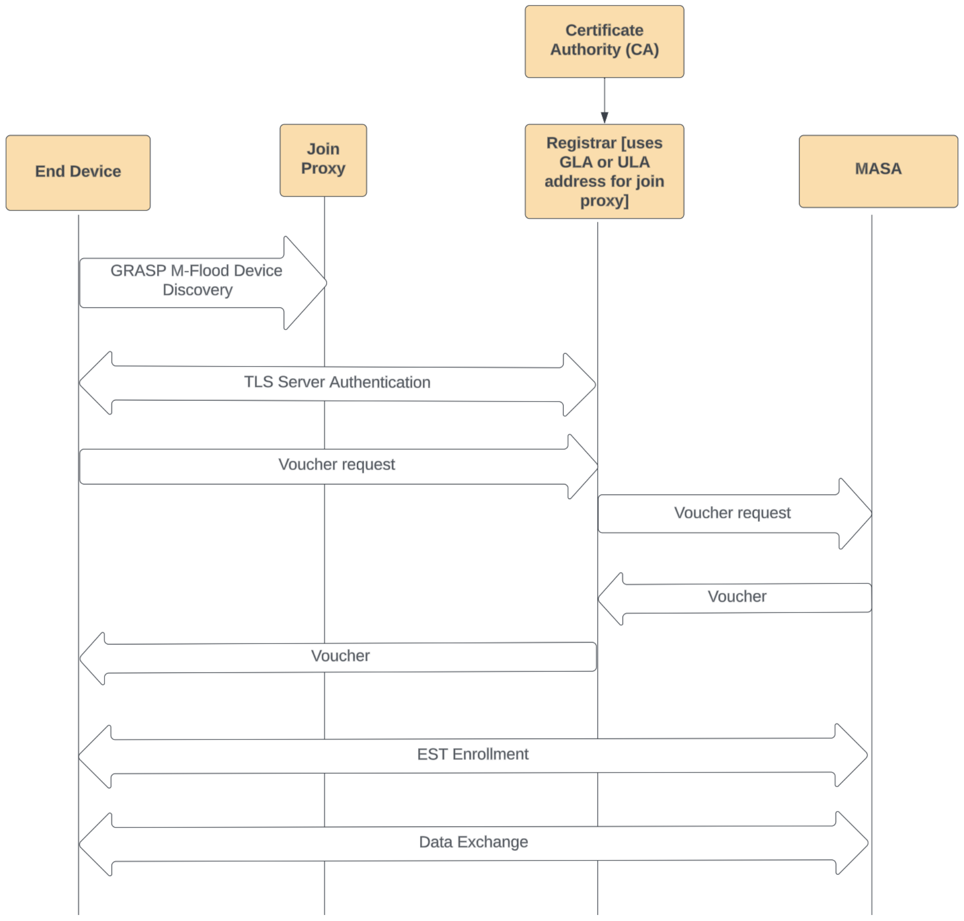

Certificate Authority (CA)
End Device
Join Proxy
Registrar [uses GLA or ULA address for join proxy]
MASA
GRASP M-Flood Device Discovery
TLS Server Authentication
Voucher request
Voucher request
Voucher
Voucher
EST Enrollment
Data Exchange

Figure 3.6: BRSKI Architecture

### 3.3.1 Message Flow in BRSKI

This section provides an in-depth examination of the various message flows within BRSKI, describing the steps and interactions between the end device (pledge), join proxy, registrar, and MASA (Figure 3.6).

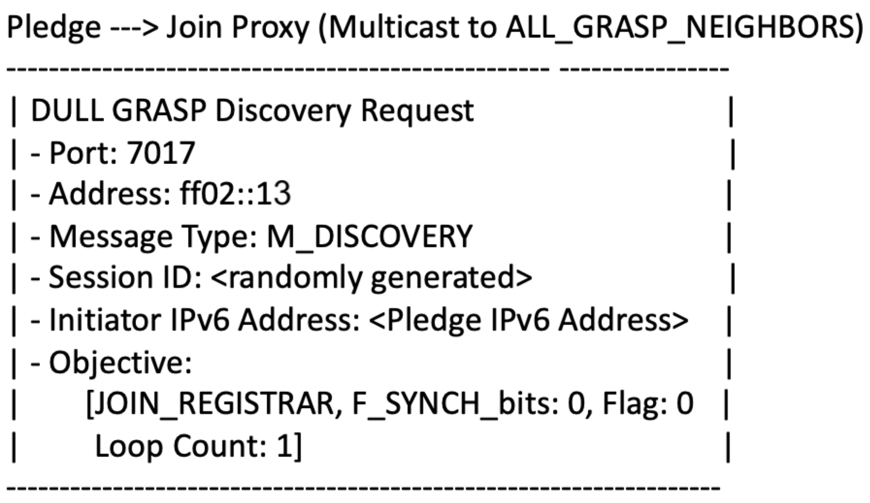
#### Discovery Request and Response

The Pledge, upon being powered on, needs to discover a local Join Proxy that will facilitate its communication with the network's Registrar. To achieve this, it sends autonomic multicast messages using Discovery Unsolicited Link-Local (DULL) [RFC8990] M_FLOOD announcements of the GeneRic Autonomic Signaling Protocol (GRASP). Below shows the DULL GRASP step by step process:

(1) Multicast Discovery Request (M _DISCOVERY): The Pledge sends a multicast Discovery Request message (M_DISCOVERY) via UDP to the GRASP_LISTEN_PORT (7017) at the link-local ALL_GRASP_NEIGHBORS multicast address on each link-layer interface. The message contains the Pledge's IPv6 address, a randomly generated session ID, and the objective to find the Join Registrar (JOIN_REGISTRAR).

discovery-message = [M_DISCOVERY, session-id, initiator, objective]

Below is a discovery-message content example(Figure 3.7):



```
Pledge ---> Join Proxy (Multicast to ALL_GRASP_NEIGHBORS)
------------------------------------------------- --------------
| DULL GRASP Discovery Request                          |
| - Port: 7017                                          |
| - Address: ff02::13                                   |
| - Message Type: M_DISCOVERY                           |
| - Session ID: <randomly generated>                    |
| - Initiator IPv6 Address: <Pledge IPv6 Address>       |
| - Objective:                                          |
|     [JOIN_REGISTRAR, F_SYNCH_bits: 0, Flag: 0         |
|      Loop Count: 1]                                   |
---------------------------------------------------------------
```

Figure 3.7: DULL GRASP Discovery Message. Surce [1].

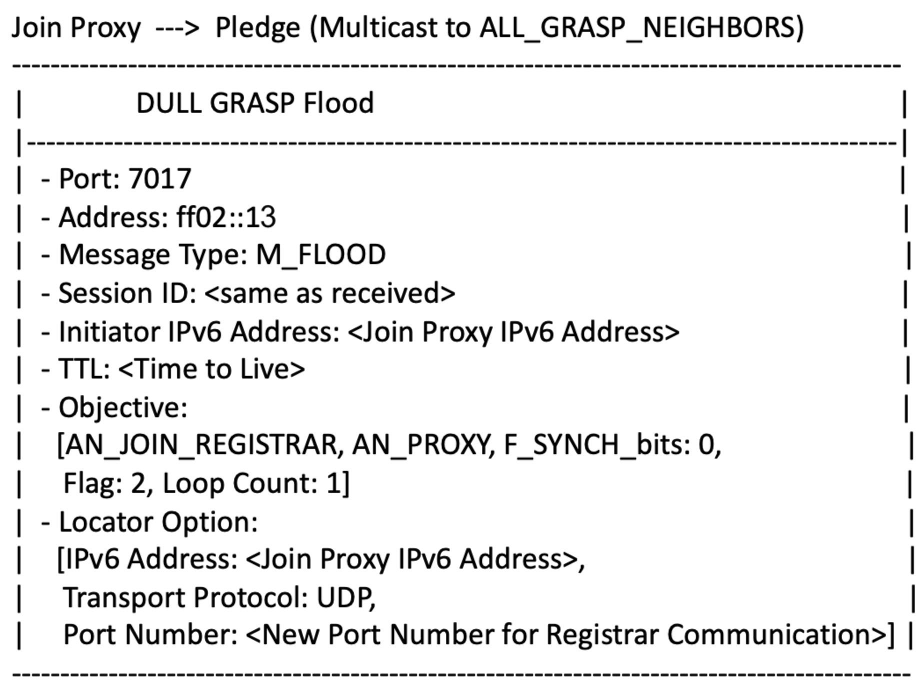
Note that the flag 0 is the indicator of the discovery message and the loop count in DULL GRASP is 1.

(2) Join Proxy Receives Discovery Request: All devices on the network listening on the GRASP_LISTEN_PORT receive the Discovery Request. The Join Proxy, upon receiving the request, prepares a response with its locator information.

(3) Multicast Flood (M_FLOOD): The Join Proxy sends a multicast Flood message (M_FLOOD) to announce its presence and provide its locator information (IPv6 address, transport protocol, port number). This message is also sent to the GRASP_LISTEN_PORT (7017) on the link-local ALL_GRASP_NEIGHBORS multicast address.

flood-message = [M_FLOOD, session-id, initiator, ttl, [objective, locator-option]]

Below is a flood-message content example(Figure 3.8):



```
Join Proxy  --->  Pledge (Multicast to ALL_GRASP_NEIGHBORS)
-------------------------------------------------------------------------------------------
|           DULL GRASP Flood                                                              |
|-----------------------------------------------------------------------------------------|
|  - Port: 7017                                                                           |
|  - Address: ff02::13                                                                    |
|  - Message Type: M_FLOOD                                                                |
|  - Session ID: <same as received>                                                       |
|  - Initiator IPv6 Address: <Join Proxy IPv6 Address>                                    |
|  - TTL: <Time to Live>                                                                  |
|  - Objective:                                                                           |
|    [AN_JOIN_REGISTRAR, AN_PROXY, F_SYNCH_bits: 0,                                       |
|     Flag: 2, Loop Count: 1]                                                             |
|  - Locator Option:                                                                      |
|    [IPv6 Address: <Join Proxy IPv6 Address>,                                            |
|     Transport Protocol: UDP,                                                            |
|     Port Number: <New Port Number for Registrar Communication>]                         |
-------------------------------------------------------------------------------------------
```

Figure 3.8: DULL GRASP Flood Message. Source [1].

(4) Pledge Receives Flood Message: The Pledge receives the Flood message and extracts the Join Proxy's locator information, storing it for subsequent secure communications.

(5) Registrar Announcement: The registrar also announces its presence using GRASP M_FLOOD messages, indicating the services it supports and the ports it listens on.

Secure Communication: The pledge uses the ACP to securely communicate with the registrar via the proxy, ensuring that all interactions are protected by the ACP's security mechanisms.

### TLS Server Authentication

In BRSKI , the TLS handshake is an essential process that establishes a secure communication channel between the Pledge (new device) and the Registrar (trusted network entity), and between the Registrar and MASA. The handshake ensures that both parties authenticate each other and agree on encryption methods to protect their communication (Figure 3.9).

The TLS handshake in BRSKI begins with the Pledge (client) initiating communication by sending a Client Hello message. This message includes the protocol version, a randomly generated number, session ID, list of supported cipher suites, and any extensions such as Server Name Indication. The Registrar (server) responds with a Server Hello, echoing back its chosen protocol version, its own random number, session ID, selected cipher suite, and extensions. Following this, the server sends its Certificate message, which includes the server's certificate chain to authenticate itself to the client. The Server Hello Done message indicates the server has finished its part of the initial negotiation.

The role of the proxy is to facilitate this communication by forwarding packets without terminating the TLS handshake, thereby maintaining the security and integrity of the connection. This type of proxy is referred to as a circuit proxy, a form of Application Level Gateway. The circuit proxy ensures that the communication stream remains intact and secure from the pledge to the registrar.

In response, the client sends a Client Key Exchange message, which includes the pre-master secret encrypted with the server's public key from the certificate. This is crucial for establishing a shared secret that will be used to generate session keys. Following this, the client sends a Change Cipher Spec message, indicating that it will start using the agreed-upon encryption for the remaining handshake messages. The Finished message is then sent, containing a hash of all previous handshake messages encrypted with the session key, verifying the client's part of the handshake.

The server replies with its own Change Cipher Spec message, indicating it too will start using encryption. It then sends a Finished message, encrypted and containing a hash of all previous handshake messages to verify the server's part of the handshake. Once both sides have verified each other's Finished messages, secure communication is established, and they can exchange Application Data encrypted with the negotiated session keys. This ensures the integrity and confidentiality of the data exchanged during the BRSKI bootstrapping process.

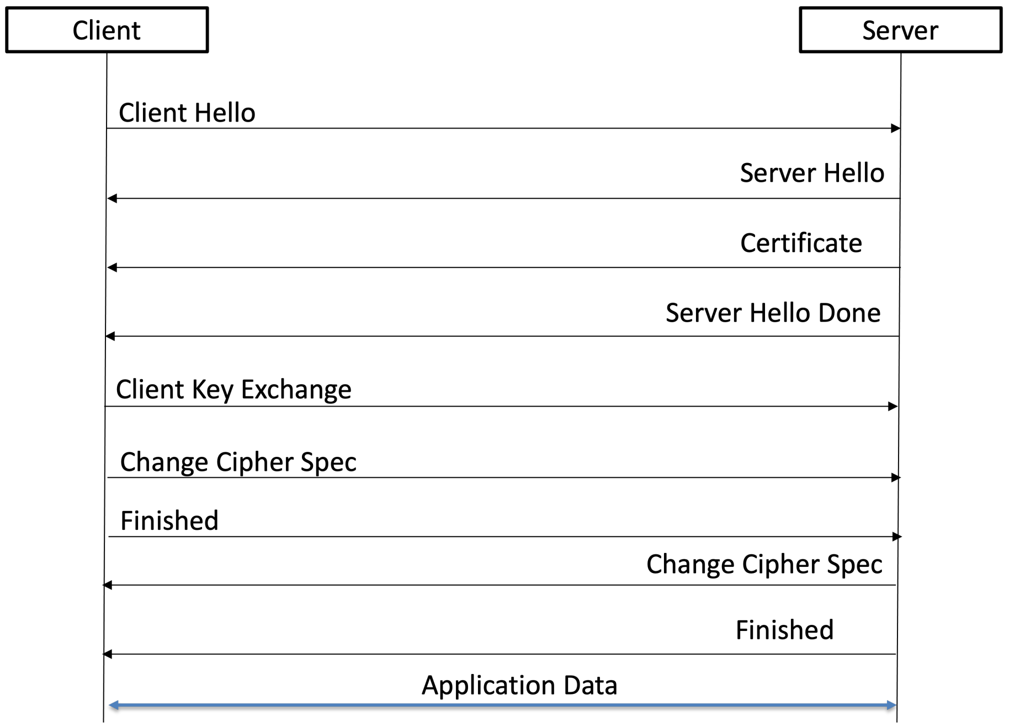


Figure 3.9: TLS Message Exchange

### Voucher Request to MASA

In BRSKI, a voucher request is a mechanism used by a device (pledge) to request a voucher from the Manufacturer Authorized Signing Authority (MASA). This process ensures the device can securely join a network. The voucher request process involves two main types of requests: the pledge voucher-request and the registrar voucher-request.

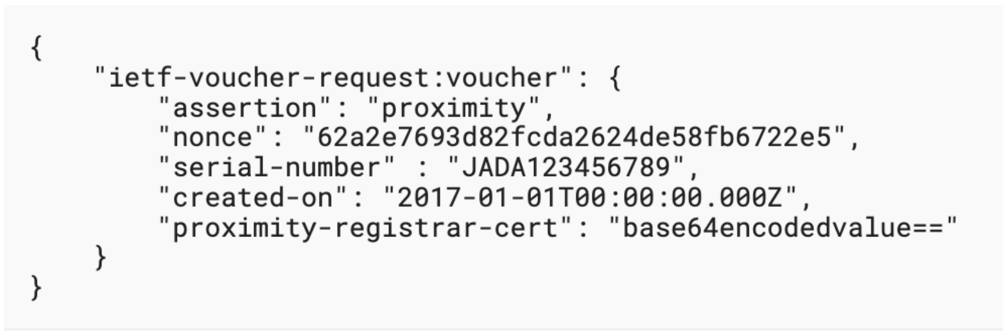
**Pledge Voucher-Request** A pledge forms the pledge voucher-request, signs it with its IDevID (Initial Device Identifier), and submits it to the registrar. The request typically includes:

- assertion: Indicates the type of proximity assertion (e.g., proximity).

- nonce: A unique value to prevent replay attacks.
- serial-number: The device's unique serial number.
- created-on: The timestamp when the voucher request was created.
- proximity-registrar-cert: The registrar's TLS server certificate, asserting proximity.

The JSON message format is chosen for its simplicity and readability, making it easier to parse, debug, and transmit over networks. Moreover, Base64 Encoding of the message ensures binary data (like certificates) can be safely included in JSON structures.

Below (Figure 3.10), is an example of JSON representation of a pledge voucher-request:



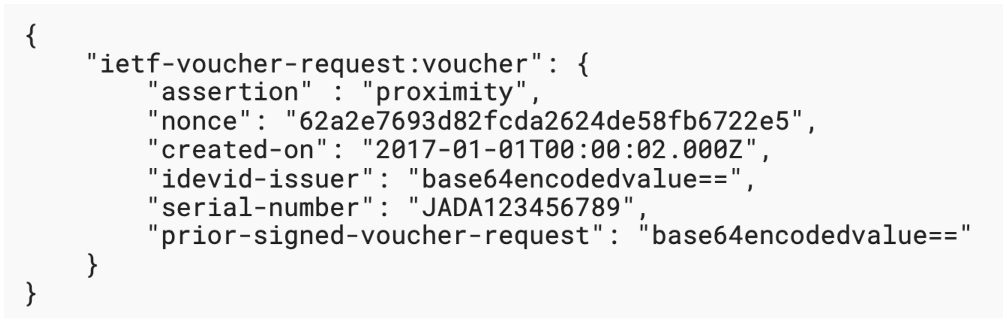
```
{
    "ietf-voucher-request:voucher": {
        "assertion": "proximity",
        "nonce": "62a2e7693d82fcda2624de58fb6722e5",
        "serial-number" : "JADA123456789",
        "created-on": "2017-01-01T00:00:00.000Z",
        "proximity-registrar-cert": "base64encodedvalue=="
    }
}
```

Figure 3.10: JSON Representation of Pledge Voucher Request. Source [1].

**Registrar Voucher-Request** The registrar takes the signed pledge voucher-request, forms its own registrar voucher-request, signs it with its registrar key pair, and submits it to the MASA. The registrar voucher-request includes:

- assertion: Carries forward the assertion from the pledge request.
- nonce: Carries forward the nonce from the pledge request.
- created-on: The timestamp when the registrar voucher-request was created.
- idevid-issuer: The issuer of the pledge's IDevID certificate.
- serial-number: The pledge's serial number, extracted from its client certificate.
- prior-signed-voucher-request: The signed pledge voucher-request, base64 encoded.

Below (Figure 3.11), is an example JSON representation of a registrar voucher-request:



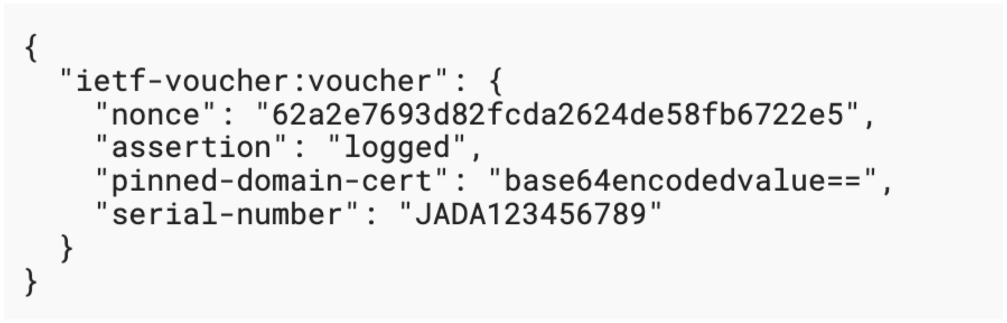
```
{
    "ietf-voucher-request:voucher": {
        "assertion" : "proximity",
        "nonce": "62a2e7693d82fcda2624de58fb6722e5",
        "created-on": "2017-01-01T00:00:02.000Z",
        "idevid-issuer": "base64encodedvalue==",
        "serial-number": "JADA123456789",
        "prior-signed-voucher-request": "base64encodedvalue=="
    }
}
```

Figure 3.11: JSON Representation of Registrar Voucher-Request. Source [1].

### Voucher Response from MASA

The MASA issues a voucher in response to a voucher-request, which is then used by the pledge to authenticate the registrar and establish a secure network connection. Upon receiving the voucher-request, the MASA evaluates it and generates a voucher. If the request is successful, the MASA responds with an HTTP 200 status code and the voucher in the response body. Otherwise, the MASA responds with HTTP 403, HTTP 404, or HTTP 415 indicating various errors such as incorrect signing, stale requests, or unsupported media types. The response contains a plain text, human-readable error message.

The voucher is a JSON object signed using CMS (Cryptographic Message Syntax) (Figure 3.12).



```
{
  "ietf-voucher:voucher": {
    "nonce": "62a2e7693d82fcda2624de58fb6722e5",
    "assertion": "logged",
    "pinned-domain-cert": "base64encodedvalue==",
    "serial-number": "JADA123456789"
  }
}
```

Figure 3.12: Example Voucher. Source [1].

Then, the registrar forwards the MASA's response to the pledge without any modifications. If the registrar accepts the pledge, the Voucher Response is sent back through the join proxy. Upon receiving a valid response, the pledge joins the domain, configures its ACP, and becomes a secure participant within the network, utilizing the certificate issued by the registrar for communication.

### Voucher Status Telemetry

In BRSKI, the voucher status telemetry is a mechanism that allows the pledge (device being onboarded) to provide feedback about the status of its voucher processing to the MASA. This telemetry data is crucial for informing the MASA whether the voucher was successfully used by the pledge or if any issues arose during the onboarding process. The primary purpose of this telemetry is to enhance security by enabling the MASA to monitor the proper use of vouchers, identify any misconfigurations, and detect potential attacks or anomalies in the onboarding process. After the pledge uses the voucher to authenticate and join the network, it sends the voucher status telemetry to the Registrar, which then forwards an audit log request to the MASA for further evaluation. An example of the voucher status telemetry is indicated in the Figure 3.13

```
{
    "version": 1,
    "status":false,
    "reason":"Informative human-readable message",
    "reason-context": { "additional" : "JSON" }
}
```

Figure 3.13: Enter Caption

Following the receipt of the audit log request, the MASA generates and send an audit log back to the Registrar. This audit log serves as a record of the voucher issuance and includes details such as the status of the voucher, the time of its use, and any detected anomalies during the onboarding process. The Registrar, and potentially the domain owner, can use this audit log to maintain a secure and verifiable trail of the onboarding process. The audit log ensures that there is a documented history of the interactions between the pledge, Registrar, and MASA, thereby supporting the integrity and transparency of the onboarding process in BRSKI. This step is crucial for confirming that the voucher was used as intended and that the device was securely onboarded into the network. The example is indicated in the Figure 3.14

```
{
  "version":"1",
  "events":[
    {
        "date":"2019-05-15T17:25:55.644-04:00",
        "domainID":"BduJhdHPpfhQLyponf48JzXSGZ8=",
        "nonce":"VOUFT-WwrEv0NuAQEHoV7Q",
        "assertion":"proximity",
        "truncated":"0"
    },
    {
        "date":"2017-05-15T17:25:55.644-04:00",
        "domainID":"BduJhdHPpfhQLyponf48JzXSGZ8=",
        "nonce":"f4G6Vi1t8nKo/FieCVgpBg==",
        "assertion":"proximity"
    }
  ],
    "truncation": {
        "nonced duplicates": "0",
        "nonceless duplicates": "1",
        "arbitrary": "2"
     }
}
```

Figure 3.14: Audit Log Response

**EST Enrollment**

With the voucher in hand, the pledge proceeds to enroll with the network infrastructure using the EEST protocol.

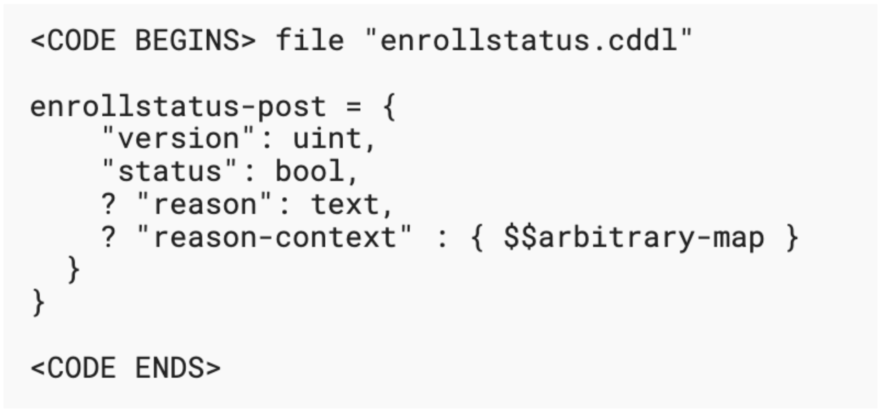
EST enrollment is integrated into BRSKI to automate the process of obtaining Certificate Authority (CA) certificates, Certification Signing Request (CSR) attributes, and client certificates. This automation reduces the need for manual intervention and simplifies device onboarding. The use of HTTP-persistent connections is recommended to simplify the state machine on the pledge and enhance security. All EST transactions occur over secure TLS connections to ensure the confidentiality and integrity of the messages. Certificates and CSRs are digitally signed to authenticate the entities involved and prevent tampering. This process involves securely exchanging certificates, keys, and other cryptographic material to establish trust and authenticate the pledge within the network.

The process begins with a CA Certificates Request, where the pledge sends a GET /cacerts request and receives a list of CA certificates in Distinguished Encoding Rules (DER) format, a binary format for encoding data structures. This is followed by a CSR Attributes Request, with a GET /csrattrs request returning the necessary CSR attributes. A CSR is a message sent from an applicant to a certificate authority (CA) to apply for a digital identity certificate. CSR Attributes are additional pieces of information requested by the CA to be included in the CSR. These attributes ensure that the certificate is generated with the necessary details and is formatted correctly for its intended use.
An example CSR might look like:

```
——BEGIN CERTIFICATE REQUEST——
MIIC... (Base64 encoded CSR)
——END CERTIFICATE REQUEST——
```

The pledge then makes a Client Certificate Request by sending a POST /simpleenroll request with a Base64 encoded CSR in the body and receives a signed client certificate in response. Finally, the pledge sends an Enrollment Status Report via a POST /enrollstatus request with a JSON payload indicating the status of the enrollment. The server responds with HTTP 200 OK on success.

To provide feedback on the enrollment status to the registrar, the pledge sends an enrollment status telemetry message. This is crucial for automated bootstrapping and lifecycle management. The enrollment status is reported via an HTTP POST request to the /.well-known/brski/enrollstatus endpoint with a JSON payload. An example payload might look like:



```
<CODE BEGINS> file "enrollstatus.cddl"

enrollstatus-post = {
    "version": uint,
    "status": bool,
    ? "reason": text,
    ? "reason-context" : { $$arbitrary-map }
  }
}

<CODE ENDS>
```

Figure 3.15: Enrollment Status POST Example. Source [1].

**Data Exchange**

Upon successful enrollment, the pledge is configured with the necessary network parameters and security policies specified in the voucher. It can now engage in secure data exchange with other network entities, enabling it to participate in network operations and communication.

### 3.3.2 Security mechanisms

BRSKI addresses key security aspects such as confidentiality, integrity, authentication, replay attack prevention, and denial of service (DoS) attack mitigation.

The voucher requests are formatted in JSON, ensuring a standardized and human-readable format for data exchange. JSON encoding rules specify that any binary content, such as certificates, must be base64 encoded. This encoding is necessary for including complex data types within JSON strings. Below shows the security mechanisms covered by BRSKI:

- Confidentiality, Integrity, and Authentication: BRSKI ensures confidentiality and integrity through TLS encryption, which protects the data exchanged between the pledge and the registrar. Moreover, the pledge presents its IDevID certificate in the TLS connection to the registrar, which includes its identity and fulfilling authentication.

- Authorization: After authenticating the pledge's identity, the registrar requests a voucher from the MASA (Manufacturer Authorized Signing Authority). The voucher is a signed statement asserting the device's authenticity and ownership. Moreover, once the pledge receives the voucher from the registrar. it validates the voucher to confirm that registrar is authorised by MASA.

- Replay and Denial of Service (DoS) Attacks:

  BRSKI implements several security metrics to ensure the integrity and availability of the system. To protect against replay attacks, BRSKI uses unique nonces and timestamps for each session, ensuring that messages cannot be reused maliciously. To mitigate DoS attacks, BRSKI implements rate limiting through mechanisms like the Retry-After header in HTTP 202 responses, instructing the pledge to wait a specified time before retrying, and by employing exponential back-off for connection retries.

  Validation checks are embedded throughout the protocol. During the TLS handshake, the pledge and registrar authenticate each other using their respective certificates. Voucher requests and responses include nonces and digital signatures to ensure freshness and integrity, with fields such as nonce, serial-number, and proximity-registrar-cert being critical for these validations. Additionally, MASA audit logs record all voucher transactions, providing an extra layer of validation that the registrar can verify to detect anomalies.

## 3.4 SCHC

### 3.4.1 On-boarding setup

SCHC relies on predefined contexts shared between communicating endpoints to compress and decompress packets effectively. These contexts must be synchronized and consistent on both ends to ensure accurate packet interpretation and data integrity. SCHC Context Initialization involves setting up the initial parameters and values that will be used for compressing and decompressing the packet headers. This includes defining the static context, which contains all the rules and parameters necessary for the SCHC operations. The context is shared between the sender and receiver to ensure consistent compression and decompression processes. Initialization ensures that both ends have a common understanding of the fields, their possible values, and how to handle them during communication [19].

Context Derivation: Contexts in SCHC are derived based on the specific network protocols and application requirements. The process involves identifying common header fields that can be compressed and establishing rules for their compression and decompression. These rules are: RuleID to find each compression rule identifier, CDA to show how to compress and reconstruct the header fields, and Field Descriptor to show the specifications of the header fields to be compressed, including their positions, lengths, and matching operators. The context is typically predefined and manually configured by network administrators.

To establish the context, protocol specifications must provide details about network protocols and their header structures to identify fields for compression. Additionally, synchronization protocols are necessary to ensure both endpoints maintain consistent contexts through mechanisms such as periodic updates or acknowledgments.

### 3.4.2 Security Considerations

As SCHC is designed to operate over LPWAN technologies, it inherently relies on the security metrics implemented by these underlying networks. SCHC by itself does not provide direct security features like encryption, authentication, or integrity protection. These must be handled by the underlying protocols (e.g., IPsec, DTLS) that SCHC helps to compress.

Chapter 4

# State of the art

In the previous chapters, we discussed the importance and necessity of transmitting IPv6 packets over LoRaWAN. We also introduced key concepts related to this topic, including autonomic networking and the onboarding process for each technology. In this chapter, we present a comprehensive review of recent research that offers solutions for communication between LPWAN and IPv6 networks, focusing on the onboarding process of LPWAN into the new IPv6 network and evaluating whether these solutions adequately address security properties, automation, and their limitations.

Finally, we present an overview of previous work on updating the SCHC context, highlighting the research gaps to illustrate the key use cases in this area of study.

## 4.1 LPWAN technologies and IPv6 integration

Among the standardized solutions used to run IP over LPWANs technologies, it is concluded in the recent studies that SCHC is a key development to facilitate this transmission [16, 23].

Aguilar et al. [24–26] focused on evaluating the performance of the SCHC framework over Sigfox, a prominent LPWAN technology, particularly emphasizing the ACK-on-Error fragmentation mode. The study primarily examined packet transfer times and the efficiency of uplink and downlink messages, offering valuable insights for IoT application design, network planning, and resource management, while also addressing the implications of varying packet sizes and error rates on transfer efficiency.

The authors in [27] focused on evaluating different Receiver Feedback Techniques (RFTs) for reliable fragmentation over LPWANs, with a particular emphasis on the SCHC framework. It explored the performance of RFTs with the List of Lost Fragments (LLF) against benchmarks. The study highlighted that the choice of RFT significantly affects performance depending on factors like error rates, packet sizes, and error patterns.

Wistuba La-Torre et al. [28] focused on evaluating the performance of SCHC over Sigfox LPWAN technology, particularly its ACK-on-Error fragmentation mode, with emphasis on packet transfer times and the number of uplink and downlink messages required. Security considerations included analyzing the impact of fragment loss rates on packet transfer time, crucial for reliable communication in IoT solutions. The study provides valuable insights for IoT solution developers, aiding in application design, network planning, and resource management, particularly in LPWAN environments where message rates are restricted.

Authors in [29] focused on developing a state machine for SCHC Fragment delivery over the Sigfox network, aiming to predict the number of uplink messages needed for successful transmission considering packet loss rates.

Muñoz et al. [10] focised on evaluating the efficiency of SCHC over LoRaWAN, particularly in terms of channel occupancy under different communication parameters such as error probability, spreading factor, and SCHC window size. The study proposes a model that accurately predicts channel efficiency based on these factors and validates it through experimental results. The key findings show that while the spreading factor and error probability significantly affect channel efficiency, the SCHC window size does not have a notable impact. This work does not discuss security considerations, onboarding methods, or human involvement in the system. Instead, it concentrates on the performance and efficiency aspects of SCHC in LoRaWAN networks.

Banti et al. [6] provide a comprehensive survey of LoRaWAN communication protocols with a focus on energy efficiency, particularly in challenging environments where power supply is limited or intermittent. The study explores how different communication protocols at the Physical, MAC, and network layers impact energy consumption, scalability, and network performance. It discusses the need for a GreenLoRaWAN communication protocol that maximizes network lifetime, enhances robustness, and addresses scalability issues. While the paper delves deeply into energy efficiency and related challenges, it does not specifically address security considerations, onboarding, or autonomous networking.

Abdelfadeel et al. [30] introduce and evaluate SCHC and its enhancement, LSCHC (Layered SCHC), designed to improve the transmission of IPv6, UDP, and CoAP headers over LPWANs like LoRaWAN. The study compares SCHC/LSCHC with IPHC (IP Header Compression) and NHC (Next Header Compression) in terms of compression efficiency, showing that SCHC/LSCHC achieves a higher compression factor, reducing transmission time and improving reliability. The work focuses on optimizing header compression for known data flows in LPWANs, though it acknowledges challenges with unknown flows.

Wistuba et al. [31] The paper discusses the implementation of the SCHC framework's fragmentation and reassembly mechanisms for LPWANs, with a focus on the ACK-on-Error mode, where acknowledgments are sent only when fragments are lost. The project aims to create an open-source, technology-agnostic implementation that initially tests in a local environment and is being extended to support Sigfox LPWAN with integration into Google Cloud Platform. The work demonstrates the potential of SCHC to enable larger payloads, improve message reliability, and add IP connectivity to constrained IoT devices.

Authors in [18] propose a methodology for evaluating the latency of IPv6 data transmission when using the SCHC protocol over LoRaWAN in real-world deployments. The study introduces a formal test procedure that maps and timestamps information flows to assess architectural delays, enabling comparisons across different SCHC implementations. The methodology was tested in various global LoRaWAN deployments, demonstrating that the IPv6 end-to-end latency with SCHC can be less than 1 second for uplink and 4 seconds for downlink in typical scenarios. The work emphasizes the practicality of the proposed approach for optimizing deployment parameters.

In the aforementioned studies, security considerations involved analyzing the reliability of SCHC Fragment delivery to ensure all fragments reach the receiver in the highly constrained LPWAN network. However, other security aspects are not analyzed in aspects of authorizing the data transmission between the two ends and prevention against possible attacks.

Authors in [32] focused on evaluating the performance of the SCHC scheme in enabling constrained

IoT devices to connect to the global Internet via LPWAN technologies, emphasizing compression and fragmentation's impact on latency, delivery ratio, and resource overhead. The device authentication is ensured by using the AES encryption standard as the security mechanism. The study emphasized the importance of balancing fragmentation sizes, channel reliability, data bandwidth, and energy consumption for optimal network performance.

Authors in [33] focused on conducting a comprehensive literature review of LoRa and LoRaWAN research from 2015 to September 2018, emphasizing technological aspects, improvements, and security considerations. The paper included a SWOT(strengths, weaknesses, opportunities and threats) analysis and highlighted remaining challenges in LoRaWAN and potential approaches to address them. it mentioned OTAA has improved some security standards in LoRaWAN. However, some vulnerabilities still exist such as DoS attack.

Authors in [34] provide a comprehensive survey of LoRaWAN, focusing on its architecture, protocol, and technologies, with an emphasis on its suitability for IoT applications. The study highlights LoRaWAN's strengths, such as low energy consumption, long-range communication, built-in security, and GPS-free positioning, while also addressing challenges like network scalability and real-time application constraints. It discusses the protocol's potential in various scenarios, including smart cities and remote monitoring. The work does cover security considerations, particularly highlighting LoRaWAN's built-in security features, but it does not specifically address onboarding or autonomic networking.

The above studies summarized in 4.1 highlight various approaches to integrating LPWAN technologies with IPv6, primarily focusing on the SCHC framework. While significant progress has been made in areas such as performance evaluation, energy consumption modeling, and security considerations, several gaps remain. Notably, most studies have concentrated on specific aspects like packet transfer efficiency, receiver feedback techniques, and the impact of SCHC on network performance. However, there is a recurring lack of comprehensive security solutions, particularly in addressing autonomic solutions, context updates in mobility scenarios, and protection against potential attacks. Additionally, the focus has often been on specific LPWAN technologies like Sigfox, with limited coverage of LoRaWAN SFs and broader security considerations necessary for reliable and secure IPv6 integration.

Authors in [35] proposes an IPv6-based architecture for LPWANs, specifically focusing on LoRaWAN, to address interoperability and security challenges in IoT deployments. It highlights the use of the SCHC (Static Context Header Compression) protocol to reduce IPv6 and UDP header overhead, enabling efficient end-to-end communication in resource-constrained networks. The work also critiques the current security mechanisms in LoRaWAN, which rely on static keys, and explores alternative key exchange protocols like EDHOC for more flexible and secure key management. The paper emphasizes the need for dynamic key updates and considers EDHOC as a suitable solution for enhancing security in LoRaWAN systems. This work explicitly addresses security considerations but does not cover onboarding or autonomic solutions.

Table 4.1: State of the art, LPWAN technologies and IPv6 integration

| Reference | Objective | Result | Not Covered | Year |
|---|---|---|---|---|
| Aguilar et al. [24] | Performance evaluation of SCHC framework over Sigfox LPWAN, specifically focusing on the ACK-on-Error fragmentation mode | Analysis of packet transfer times, efficiency of uplink and downlink messages | Security considerations not covering LoRaWAN SFs no autonomic solution | 2021 |
| Aguilar et al. [27] | Comparative analysis of Receiver Feedback Techniques (RFTs) | Examination of the impact of RFT selection on network performance metrics | Security considerations no autonomic solution | 2021 |
| Ertürk et al. [34] | Review and analyze LoRaWAN's architecture, protocol, and applicability in IoT scenarios. | Identified strengths like low energy consumption, long-range communication, and built-in security, along with challenges in scalability and real-time application support | Security considerations no autonomic and Onboarding solution | 2021 |
| Wistuba La-Torre et al. [28] | Development of a state machine for SCHC Fragment delivery over Sigfox network to predict the number of uplink messages | Security analysis to ensure reliable SCHC Fragment delivery | Security considerations autonomic solution | 2022 |
| Aguilar et al. [29] | Provision of an energy consumption model for Sigfox LPWAN utilizing SCHC | relationship between SCHC Packet size and energy consumption, highlighting decreased transfer efficiency with larger SCHC Packet sizes | Security considerations no autonomic solution | 2022 |
| Muñoz et al. [10] | Evaluate channel occupancy efficiency of SCHC over LoRaWAN under various communication parameters. | Predicts efficiency, showing that spreading factor and error probability influence performance, while window size does not | Security considerations, onboarding, autonomic networking | 2022 |
| Sanchez-Gomez et al. [32] | Evaluation of SCHC scheme's performance in LPWAN connected to IPv6 | Analysis of latency, delivery ratio, resource overhead, and security considerations regarding AES Encryption | no solution against possible attacks no autonomic solution not mentioning about context update in mobility | 2020 |
| Haxhibeqiri et al. [33] | A literature review on LoRaWAN with security considerations | OTAA has security standard improvements in LoRaWAN but still vulnerable against DoS attacks | Authentication, Integrity, and solution to prevent from possible attacks is not covered no autonomic solution | 2018 |
| Gomez et al. [16] | SCHC as a solution to connect LPWAN to IPv6 network | Analyze SCHC on different LPWANs | no Security considerations not covering LoRaWAN SFs | 2020 |

## 4.2 On-boarding solutions

Recent studies have indicated that as the number of IoT devices grows, manual device onboarding by humans becomes impractical. Additionally, in some use cases, IoT devices are not accessible for updates or reconfiguration by humans. For instance, Alvarez et al. [36] discussed the deployment of IoT devices on ground and satellite. This paper focused on IoT device deployment where the terrestrial infrastructure is not feasible or accessible. Another example is provided by Gomez et al. [16] who highlighted that LPWANs enable industrial IoT applications by covering thousands of IoT devices using the star topology. However, the capacity constraints of LPWANs challenge IPv6 and even 6LoWPAN solutions, prompting the introduction of the SCHC mechanism and Fragmentation by IETF to address these issues.

Recent studies have identified autonomic networking as the most effective solution to address the challenge of human access. The following works discuss these studies in detail.

Arzo et al. [37] provide a survey on network automation in IoT and its advantages. One of the advantages is self-protection, which is related to security against attacks such as DoS attacks. However, this paper did not mention about device authentication and secure on-boarding. Zavala, et al. [38] aim to address challenges in IoT systems, particularly focusing on achieving full autonomic behavior. By proposing new architectural and functional blocks, the paper enables self-configuration, a key aspect of managing IoT systems at scale, utilizing a self-management algorithm based on Utility Theory. Results show that the proposed framework can effectively manage IoT systems without human intervention, even with low-cost and limited processing capacity devices, thereby addressing scalability and diversity challenges in the IoT landscape.

Sari et al. [39] review recent studies on network communication protocols supporting the autonomic Internet of Things (IoT), focusing on properties like self-organization, self-optimization, self-protection, and self-energy-awareness. The authors identify energy efficiency as a major concern and highlight the incorporation of energy harvesting awareness into protocol designs, particularly at the MAC layer. The review maps protocol designs according to supported autonomic properties and emphasizes the potential of bio-inspired algorithms for addressing complexity in IoT networks. The paper suggests future research opportunities to enhance IoT infrastructure in terms of resource constraint, heterogeneity, mobility, scalability, and security through intelligent and autonomic means. Tahir et al. [40] underscore the impracticality of manually managing the vast number of devices in the Internet of Things (IoT) ecosystem and advocate for autonomic computing to minimize human intervention. While acknowledging the current limitations and challenges, the paper emphasizes the importance of progressing toward achieving true autonomy in IoT through intelligent algorithms and automated procedures, offering insights to guide future research directions in autonomic computing for IoT.

The following works are among the few studies that have focused on constrained networks within the context of autonomic networking.

Richardson et al. [22] defined the Constrained Bootstrapping Remote Secure Key Infrastructure (cBRSKI) protocol, offering a solution for secure zero-touch onboarding of resource-constrained IoT devices into a network. The outcome is the proposed specification of cBRSKI, a variant of the BRSKI protocol, which uses a compact CBOR-encoded voucher, EST-over-CoAPS, and DTLS-secured CoAP (CoAPS) for secure communication, thus addressing the unique challenges of secure enrollment in constrained networks with limited throughput and resource constraints. Their proposal assumes the constrained network is already IP-enabled; therefore, there is no consideration about the need of SCHC. In [41], Afzal et al. investigated CoAP, SCHC, and CBOR schemes to enhance communication efficiency, demonstrating significant reductions in packet sizes compared to HTTP and JSON.

The studies summarized in Table 4.2 provide a comprehensive overview of various onboarding solutions for IoT devices, particularly focusing on autonomic networking and secure enrollment in constrained networks. While significant advancements have been made in enabling self-management and reducing human intervention, several gaps remain. In those solutions the use of SCHC for connecting LPWANs to IPv6 networks is explored but the security considerations or specific challenges related to LoRaWAN SFs are not addressed. Although a survey in autonomic networking in IoT was published, identifying challenges and opportunities, but it lacked solutions for authentication and secure onboarding. Also, in the framework proposed by Zavala et al. [38] for full autonomic behavior in IoT systems, enabling management without human intervention, yet the security considerations are omitted. Lastly, Richardson et al. [22] introduced cBRSKI for secure zero-touch onboarding in constrained networks but did not cover how to compress messages for LPWAN to IPv6 networks or how to share the SCHC context. These studies collectively highlight the progress and existing challenges in achieving secure and autonomic onboarding of IoT devices, particularly in resource-constrained environments where SCHC is used to interoperate with IP networks.

Table 4.2: State of the art, Onboarding solutions

| Reference | Objective | Result | Not Covered | Year |
|---|---|---|---|---|
| Gomez et al. [16] | SCHC as a solution to connect LPWAN to IPv6 network | Analyze SCHC on different LPWANs | no Security considerations not covering LoRaWAN SFs | 2020 |
| Arzo et al. [37] | a survey on autonomic networking in IoT and protection against attacks | identify different challenges and opportunities in automation. | no authentication solution and secure on-boarding | 2021 |
| Zavala et al. [38] | propose full autonomic behavior in IoT systems | enable IoT systems to be managed without human intervention even in constrained devices | no security consideration | 2019 |
| Richardson et al. [22] | a solution for secure zero-touch onboarding of resource-constrained IoT devices into a network using CBOR and COAP | Addressed the unique challenges of secure enrollment in constrained networks with limited throughput and resource constraints | does not mention how to compress messages from LPWAN to IPv6 networks and how to share context | 2024 |

## 4.3 SCHC context management - Use cases and solutions

This section outlines the use cases that illustrate the need for a SCHC context management scheme and the corresponding solutions proposed in previous studies.

Moons et al. [42] highlighted the challenges that arise as IoT devices, now equipped with multiple LP-WAN radio technologies, require a unified protocol stack that is independent of the underlying technology. A key challenge is the mobility and roaming of IoT devices, particularly how packet header values, such as source and destination IP addresses, change when devices transition between different networks like NB-IoT and LoRaWAN. This work emphasizes the necessity for a solution to manage network heterogeneity and maintain seamless communication. The authors propose that, while current SCHC configurations are static, future implementations should support dynamic contexts to ensure communication efficiency as devices move across different networks.

Another use case in which the SCHC context needs to be updated is to reflect changes in network architecture and infrastructure. For example, when new routers or gateways are added to an IoT network, it may necessitate updates to the SCHC context to reflect new routing header fields, ensuring that devices can adapt to new routing paths [43]. In the research provided by Minaburo et al. [19], the authors discussed the scenario in which recent application layer protocols, such as CoAP, are updated with new header options. They emphasize that, if CoAP introduces a new option, the SCHC rules should be updated with new rule IDs for header compression to ensure efficient communication.

The architecture shown in Fig. 4.1, illustrates an example environment where some of the aforementioned use cases may take place, e.g., a device changing the LPWAN technology in use, from NB-IoT to LoRaWAN, or the introduction of new header options that would affect the compression of protocols like CoAP.

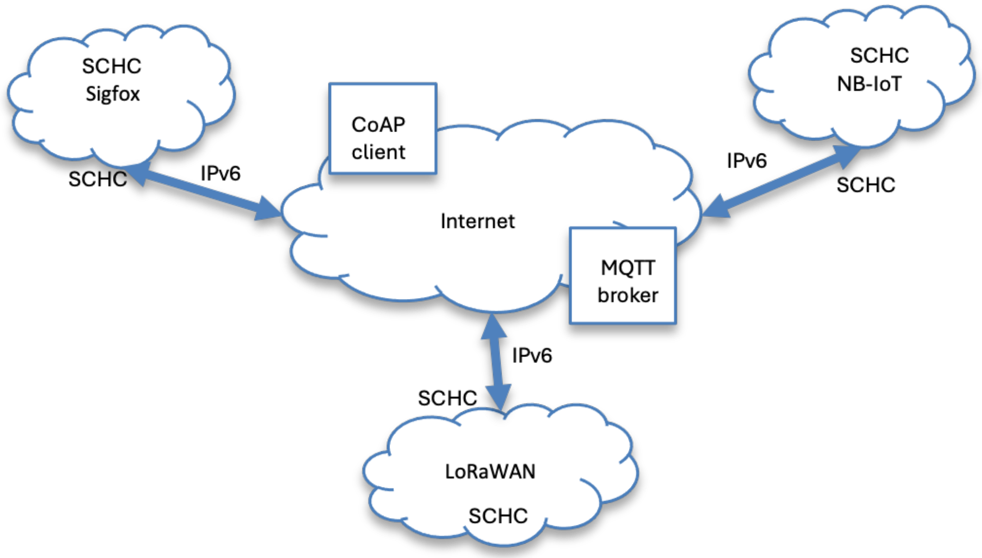


Figure 4.1: Dynamic Context Update Use Cases

The studies summarized in Table 4.3 provide a comprehensive overview of previous approaches to the problem of dynamic context management in SCHC. The Dynamic Context Header Compression (DCHC) protocol, proposed by Ayoub et al. [44], defines a dynamic management and real-time SCHC context update mechanism. However, ensuring synchronization between the Mobility Management Server (MMS) and end devices might be challenging, especially in highly dynamic environments with frequent handovers and network changes. Inefficient context updates could lead to increased latency or packet losses if not managed efficiently. Additionally, continuous monitoring and context reconfiguration may impose significant computational and communication burdens on the network infrastructure [32].

Moons et al. [17] provided a system for managing dynamic SCHC contexts. However, they do not support different spreading factors in a LoRaWAN environment, leading to problems as their fragmentation-free solutions may exceed the MTU allowed in certain spreading factors. They also fail to consider LoRaWAN's device classes and their limitations in sending multiple uplink messages. Another proposed solution is the

DNS-based approach provided by Bernard et al. [45], which dynamically finds the context rules associated with an end device and downloads them from an HTTP server. However, this method introduces query overhead when handling a large number of devices.

Table 4.3: State of the art, SCHC context management solutions

| Reference | Objective | Result | Not Covered | Year |
|---|---|---|---|---|
| Moons et al. [42] | Address challenges in IoT mobility with multiple LPWAN technologies | Highlighted the need for dynamic SCHC context updates | No dynamic management across different SFs and LoRa classes | 2019 |
| Ayoub et al. [44] | Propose the Dynamic Context Header Compression (DCHC) protocol | Introduced dynamic and real-time SCHC context management | Synchronization challenges in dynamic environments; computational burdens. | 2020 |
| Bernard et al. [45] | Propose a DNS-based approach for dynamic SCHC context rule retrieval. | Enabled context rule updates via HTTP servers. | Query overhead with many devices; updates limited to end devices, not gateways. | 2020 |
| Abdelfadeel et al. [43] | Address SCHC context updates due to changes in network architecture. | Emphasized the need for context updates when network elements change. | No comprehensive strategies for managing dynamic routing paths | 2018 |
| Moons et al. [17] | Develop a system for dynamic SCHC context updates | Provided a solution for dynamic context update | Not covering LoRaWAN fragmentation, different classes, and downlink limitations. | 2021 |

Although there has been progress in the state-of-the-art about context management for SCHC-enabled IoT networks, several gaps remain in the existing solutions. For instance, while dynamic context updates are proposed, there is a lack of comprehensive mechanisms for managing these updates in real-time, especially in highly dynamic environments. Existing solutions like the Dynamic Context Header Compression (DCHC) protocol and DNS-based approaches introduce challenges such as increased latency, computational burden, and overhead in query handling, particularly when dealing with a large number of devices, which will affect constrained deployments. Moreover, some of these solutions focus on updating the context at the end device but neglect the SCHC gateway, leading to potential synchronization issues [43, 45]. Additionally, current approaches do not adequately consider the different spreading factors of LoRaWAN, which can lead to the need of fragmentation. Furthermore, essential aspects like security, particularly in the context of device mobility, remain under-explored, with insufficient support for integrity, authentication, and authorization in the exchange of SCHC rules, leaving networks vulnerable to unauthorized access. Addressing the aforementioned gaps is essential for achieving a secure, scalable, and autonomic SCHC context network management, as it will be discussed in the next chapters.

# Chapter 5

# Problem Statement

In the previous chapters, we emphasized the growing importance of IPv6 networks and the need to ensure LoRaWAN compatibility for communication with IPv6. Additionally, we discussed the need of managing SCHC contexts updates in response to possible network configuration changes. Once a change in the network triggers updates that may affect the effectiveness and efficiency of the SCHC compression/fragmentation operation, we are faced with two options:

- Option 1: The SCHC entities continue compression only with the rules that remain valid after the change. However, this will reduce the efficiency of the SCHC compression, with an increased SCHC packet size that may trigger the need of fragmentation. With fragmentation, the transmission process will increase in complexity and will need additional downlink transmissions for ACK packets, in networks that are already highly constrained in the downlink.
- Option 2: The SCHC context is updated. However, the updating process has not been considered in the SCHC standard [19], where the static property was assumed permanent. We argue that the context may not change frequently but may not be permanent either, as demonstrated by the use cases discussed in the previous chapter. Furthemore, after reviewing the existing solutions, it was concluded that none of the previous works fully provides a secure and autonomic system for onboarding and SCHC context management over an integrated LoRAWAN/IPv6 network.

To solve the stated problem, we are required to:

- Define an architecture in which a LoRaWAN end device: 1) is securely onboarded to the Internet with a globally-routable IPv6 address; and 2) has a secure mechanism to manage SCHC context updates. When doing this, we should ensure that we take advantage of any feature of the LoRaWAN architecture that simplifies the necessary exchanges, such as the existing security protocols, and data transmission methods.
- Demonstrate that the defined architecture operates in a way that is feasible given the technology parameters (e.g., a variable MTU due to changes in the spreading factors of LoRaWAN), and the delay accepted during setup of typical IoT applications.

In the following, we introduce the hypotheses, detailing the anticipated outcomes of our solution. Then, we outline the proposed objectives that will help demonstrate the hypotheses. Finally, we identify the challenges and considerations for the solution design.

## 5.1 Hypothesis

The integration of SCHC, with an initial set of rules, and secure onboarding via BRSKI, will enable the development of an architecture that achieves the following goals:

- **Goal 1**: Efficiently and securely obtain a globally-routable IPv6 address for the LoRaWAN end device, utilizing the SCHC adaptation layer for interactions with IP entities.
- **Goal 2**: Establish a secure management channel for SCHC context updates, allowing dynamic rule adjustments to support the device's tasks while ensuring unique device identification.
- **Goal 3**: Maintain reasonable access times despite the additional overhead introduced by BRSKI and cBRSKI within the management plane.

## 5.2 Objectives

To achieve these goals, the research will focus on the following objectives:

(1) Design an onboarding process for the LoRaWAN end device to obtain a globally-routable IPv6 address, ensuring efficient and secure communication with the IPv6 network while handling constrained bandwidth.

(2) Develop a mechanism to establish a secure management channel for SCHC context management, enabling dynamic rule updates through autonomic solutions like BRSKI and cBRSKI, while ensuring the device's unique identification.

(3) Implement and evaluate the proposed system architecture to assess the impact of the additional overhead on access times, focusing on the performance of BRSKI and cBRSKI in the management plane.

## 5.3 Challenges and considerations

The solution architecture should be structured into three layers to meet the outlined goals:

- User layer: Emulation of a Secure Link using LoRaWAN plus SCHC. This layer focuses on establishing a secure communication link by integrating LoRaWAN with SCHC.
- Management layer: Autonomic Management of the Device within a *Management Domain*. In this layer, the device is managed automatically. The key elements of this layer include:
  - The end device has a verifiable identity within the management domain. LoRaWAN is a closed system which means, it does not typically interact directly with external systems or networks. The first step is to assign a globally-routable IPv6 address, valid in an open system, such as the new IPv6 network.
  - There is mutual authentication that ensures that both the End Device and the Registrar authenticate each other.
  - There is confidentiality to protects the data being transmitted from unauthorized access.
  - There is integrity to ensures that the data remains unchanged and trustworthy throughout transmission.

    - Although the LoRaWAN OTAA configuration provides essential security features, such as mutual authentication, origin authentication, integrity protection, encryption, and replay protection, these measures are confined to the LoRaWAN environment. They do not address the additional authorization required for the ED to join and operate securely within the IPv6 network.
    - The autonomic management approach will streamline the device's operation within the network, making it more efficient and secure.

- Emulation layer: the system provides a suitable North-Bound interface, which involves creating a secure and controlled interface that allows applications to access the device. The interface must be designed to ensure that all interactions with the device are secure, providing the necessary controls to maintain the integrity and confidentiality of the data and operations.

# Chapter 6

# Proposed Solution

In the previous chapters, we discussed the key concepts of onboarding LoRaWAN into an IPv6 network and the role of autonomic networks, which form the foundation of this research. The focus of this study is to establish a seamless connection between LoRaWAN, a constrained network, and IPv6, a non-constrained environment, within the framework of an autonomic network infrastructure.

To achieve this, the proposed solution is divided into three main components: LoRaWAN onboarding to create an IPv6 secure link, BRSKI to create an API, and ACPs to provide APIs for upper-level use. Each of these components is crucial for the overall system and builds upon the previous one.

## 6.1 LoRaWAN onboarding to create IPv6 secure link

The first step in the proposed solution is to onboard LoRaWAN devices into the IPv6 network by creating a secure link. This involves configuring the LoRaWAN device and ensuring that the SCHC context required for autonomic configuration is pre-stored on both the end device and the SCHC gateway. This setup is essential to ensure that IPv6 packets can be efficiently transferred over the LoRaWAN network during the onboarding process, facilitating a smooth and secure integration.

Figure 6.1 illustrates a state diagram of the overall system. The state descriptions are as follows:

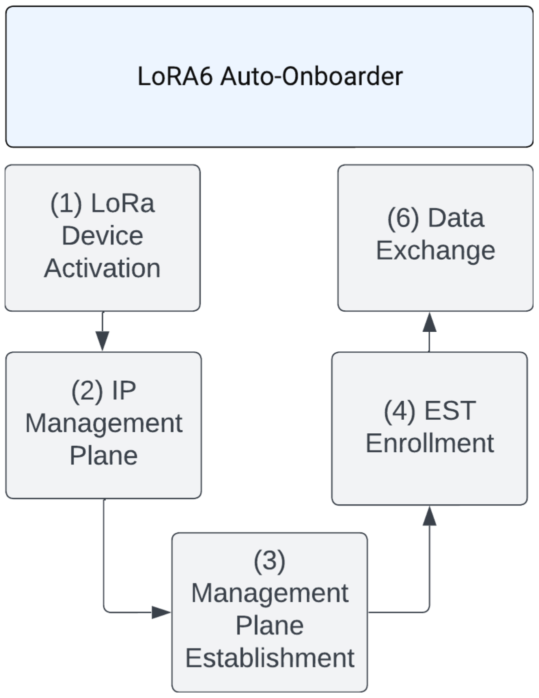


Figure 6.1: LoRA6 Auto-Onboarder

(0) Pre-configure SCHC context on LoRa and the SCHC Gateway

(1) Configure LoRa Device in a LoRaWAN network.

(2) Auto configuration of Link Layer IPv6 address.

(3) BRSKI on-boarding step

(4) This step is the end of management plane and the start of Data Plane to share messages between the LoRa End Node and the IPv6 Destination Host. To do so, a new SCHC context is required to be configured.

(5) In this stage, the system is ready to exchange data.

The following, shows the protocol stack of each step in this On-boarding system.

In the first step, LoRa is configured in the LoRaWAN network using Over-The-Air Activation (OTAA). The end device transmits data from the LoRa physical layer to the LoRaWAN network through the LoRaWAN gateway. As illustrated in Figure 6.2, the User IP layer started here is the network server IP. Depending on the backend implementation, if a reliable transport layer is needed, the TCP protocol is used over the IP layer; otherwise, the UDP protocol is employed. Data security is ensured with AES encryption between the end device and the network server, as well as between the end device and the application server.

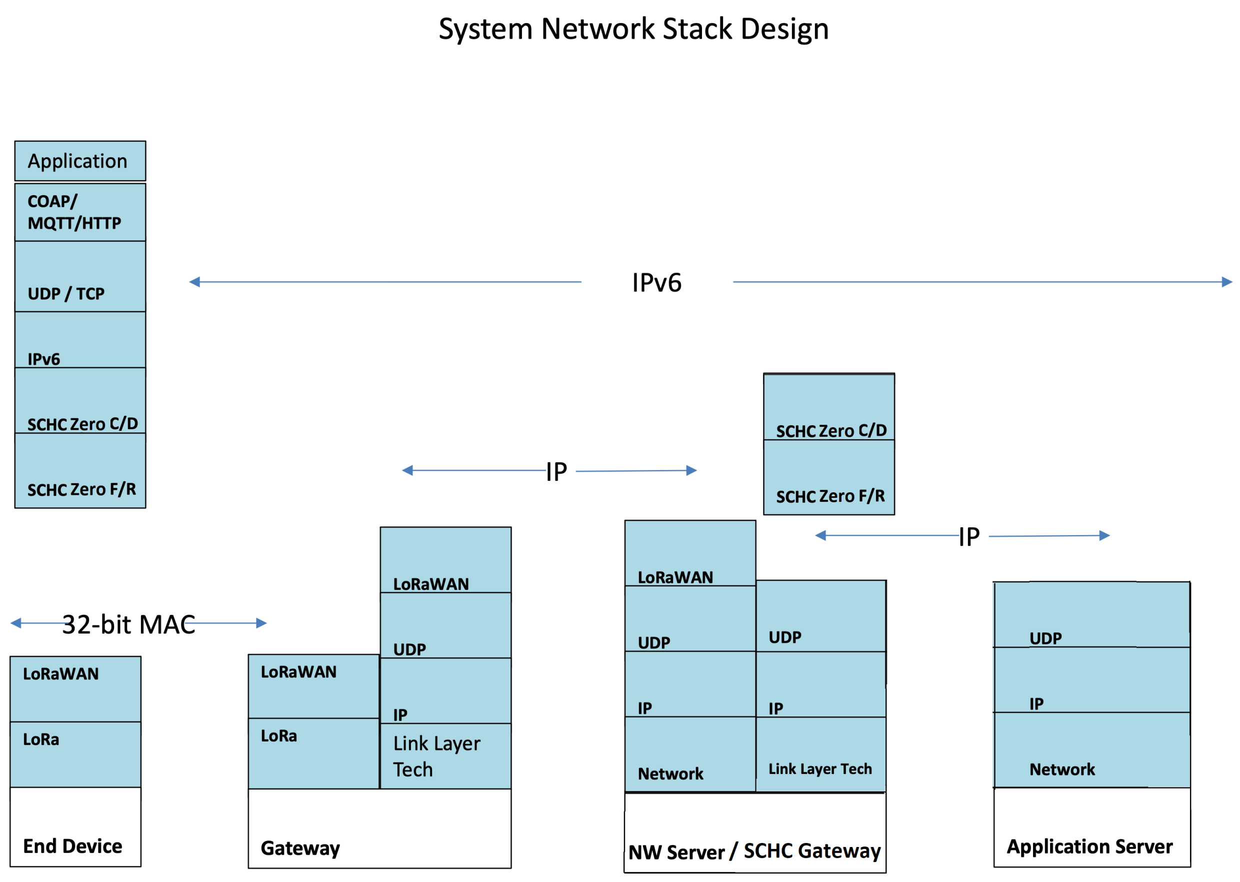


Figure 6.2: System Network Stack Design LoRaWAN Onboarding

The Figure 6.2 illustrates the system network stack design for LoRaWAN onboarding, highlighting the layered protocol architecture across different network components, including the End Device, Gateway, Network Server, and Application Server. The stack begins at the End Device with LoRa and LoRaWAN at the physical and MAC layers, progressing through SCHC for both C/D and F/R, and then onto IPv6, UDP/TCP, and application protocols like COAP, MQTT, or HTTP. The Gateway and Network Server follow a similar stack, facilitating the transmission of data through the LoRaWAN network and onto the IP network. Once connected to the LoRaWAN network, the management plane is initiated, establishing a secure connection with the ANI registrar to obtain an ANI certificate, which enables secure communication with the broader network. The image effectively demonstrates the seamless integration of LoRaWAN with IP-based protocols, ensuring secure and efficient onboarding and data exchange within the network.

## 6.2 BRSKI to create API

Building on the secure link established in the previous section, the next step is to use BRSKI to create an API that facilitates secure onboarding and communication.

The initial step for the device to connect with the registrar involves obtaining a link-local IPv6 address as showing in the Figure 6.3. The End Device creates its own link-local IPv6 address during SLAAC. Subsequently, the End Device can connect to the Circuit Join Proxy using Discovery and Flooding messages.

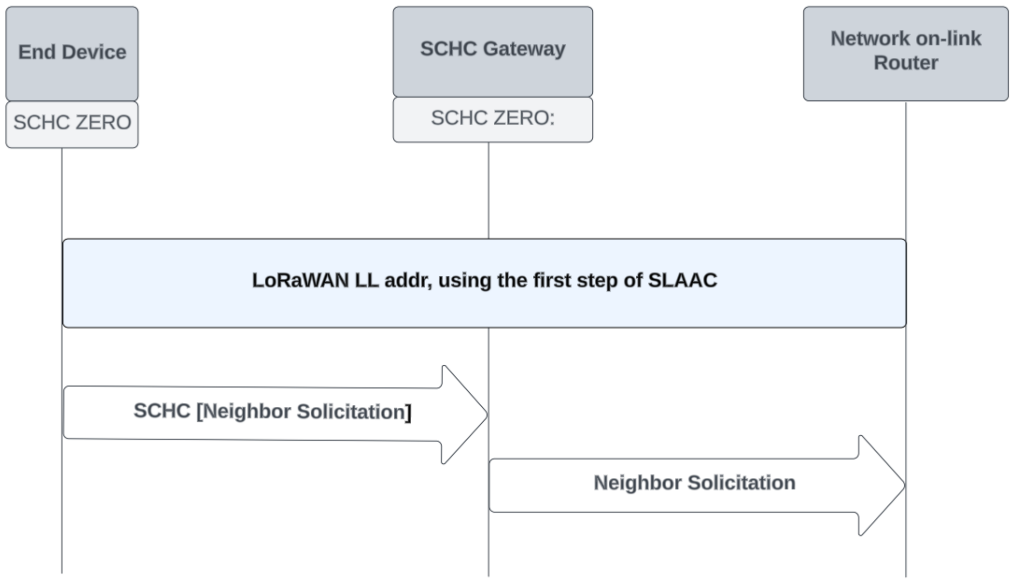


Figure 6.3: IPv6 link local address configuration sequence diagram

The Secure IP management Plane includes the DULL GRASP Mflood messages to locate the registrar; TLS handshake to authorize the End Device for the Registrar and voucher request and response to authorize the registrar for the End Device, and finally, the voucher status telemetry to indicate the voucher verification is received by the ED. The detailed messages are indicated as sequence diagrams in the Figures 6.3, 6.7, and 6.8.

Figure 6.4 illustrates the protocol stack at this stage, omitting components between the end device and the Application Server for clarity. The figure focuses on upper layers, excluding the physical layer.

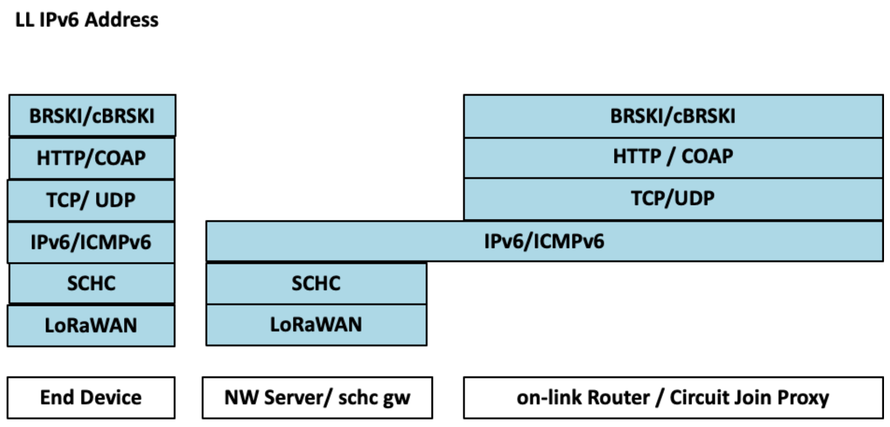


Figure 6.4: System Network Stack Design IP management plane onboarding

The application layer utilizes BRSKI over HTTP/TCP or cBRSKI over CoAP/UDP for constrained networks.

According to the RFC8990 [13], DULL GRASP M-Flood and M-Response messages are sent over UDP protocol for the purpose of initial neighbor discovery during the ACP construction phase. Once the ACP is established, GRASP flood and response messages typically use TCP for reliable transport within the ACP or UDP. Note that if using unreliable transport like UDP for those messages, GRASP must ensure no IP fragmentation occurs. Messages must be limited in size to avoid fragmentation, adhering to the IPv6 packet size limit of 1280 bytes unless a larger minimum link MTU is known (Figure 6.5).

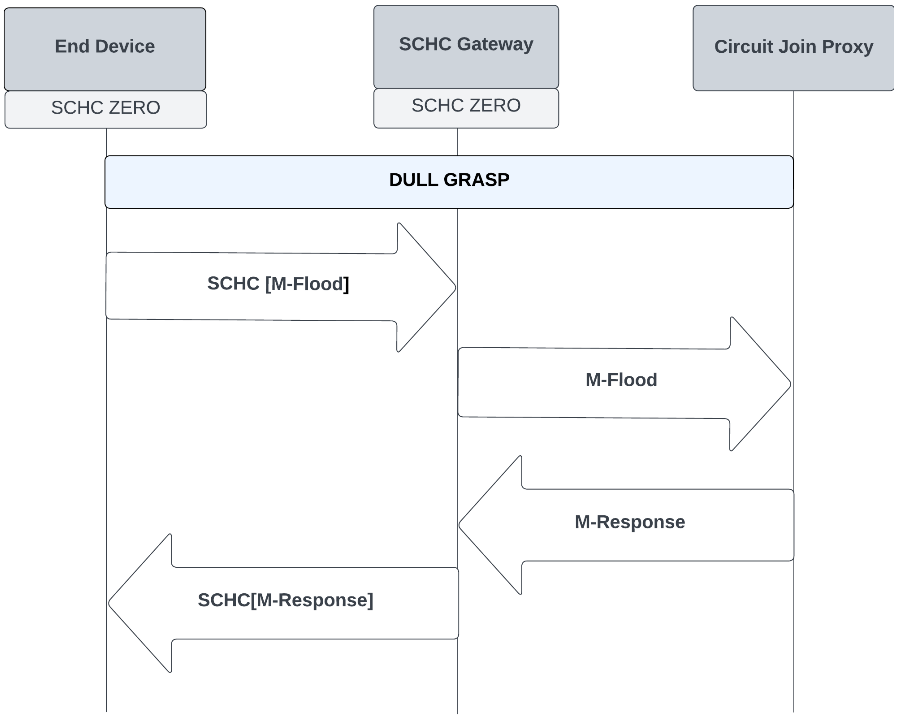


Figure 6.5: Locating registrar with DULL GRASP

The Figure 6.6 illustrates the process by which the Registrar is located through Discovery messages. Following this, a TLS handshake is conducted between the End Device, the Registrar, and the MASA, with the messages being relayed through the Circuit Join Proxy. The Circuit Join Proxy operates as an Application Level Gateway (ALG) [46], serving as a stateful proxy that facilitates communication across different network address realms. It accomplishes this by transparently relaying the TLS handshake messages without terminating or inspecting them, thereby ensuring secure communication between the devices and the Registrar/MASA without altering the TLS handshake process.

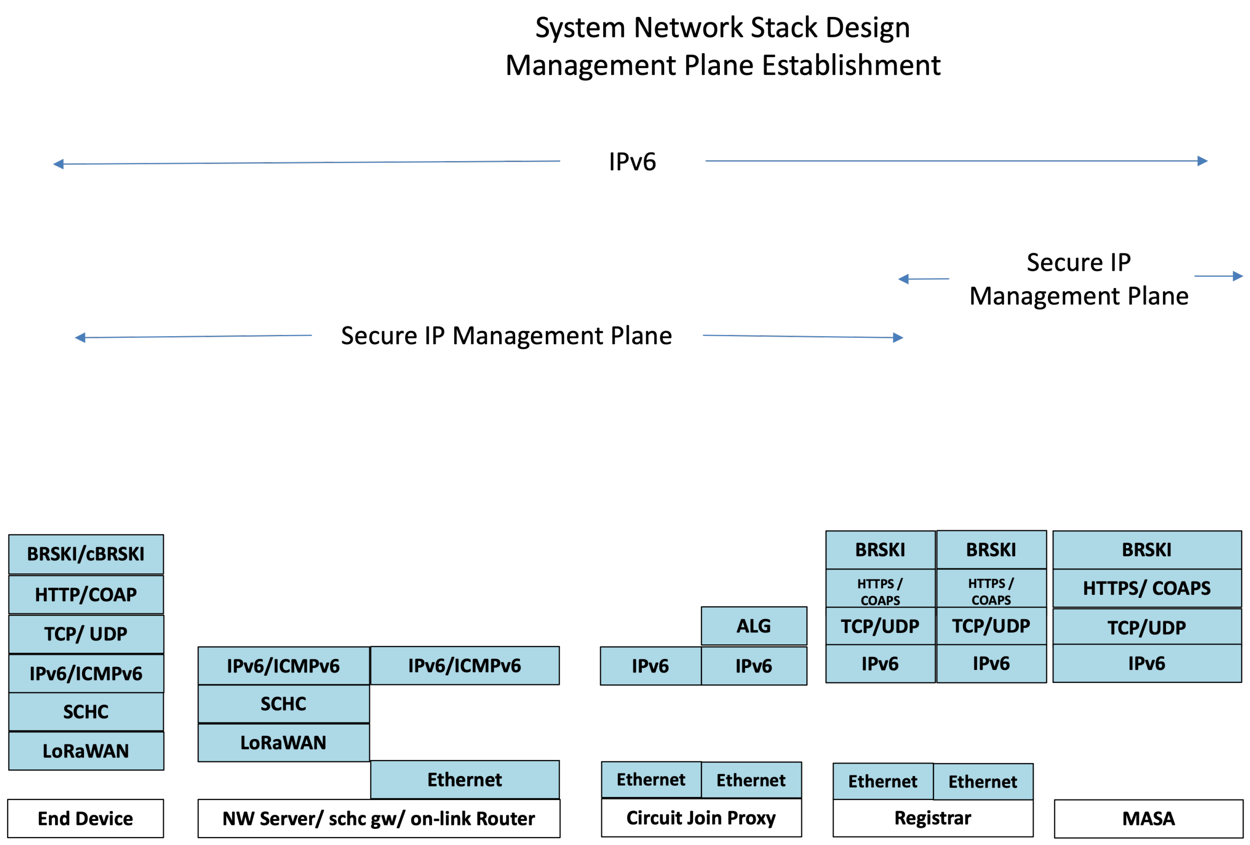


Figure 6.6: System Network Stack Design Management Plane Establishment

In this stage the stack shows the use of BRSKI with HTTPS over TCP or cBRSKI with COAPS over UDP for secure message exchange, emphasizing the layered protocol integration, from LoRaWAN and SCHC at the End Device level to higher-level protocols like IPv6, TCP/UDP, and HTTP/COAP, as the communication flows through the network infrastructure. This setup ensures a secure and efficient onboarding process for IoT devices within an IPv6-based network.

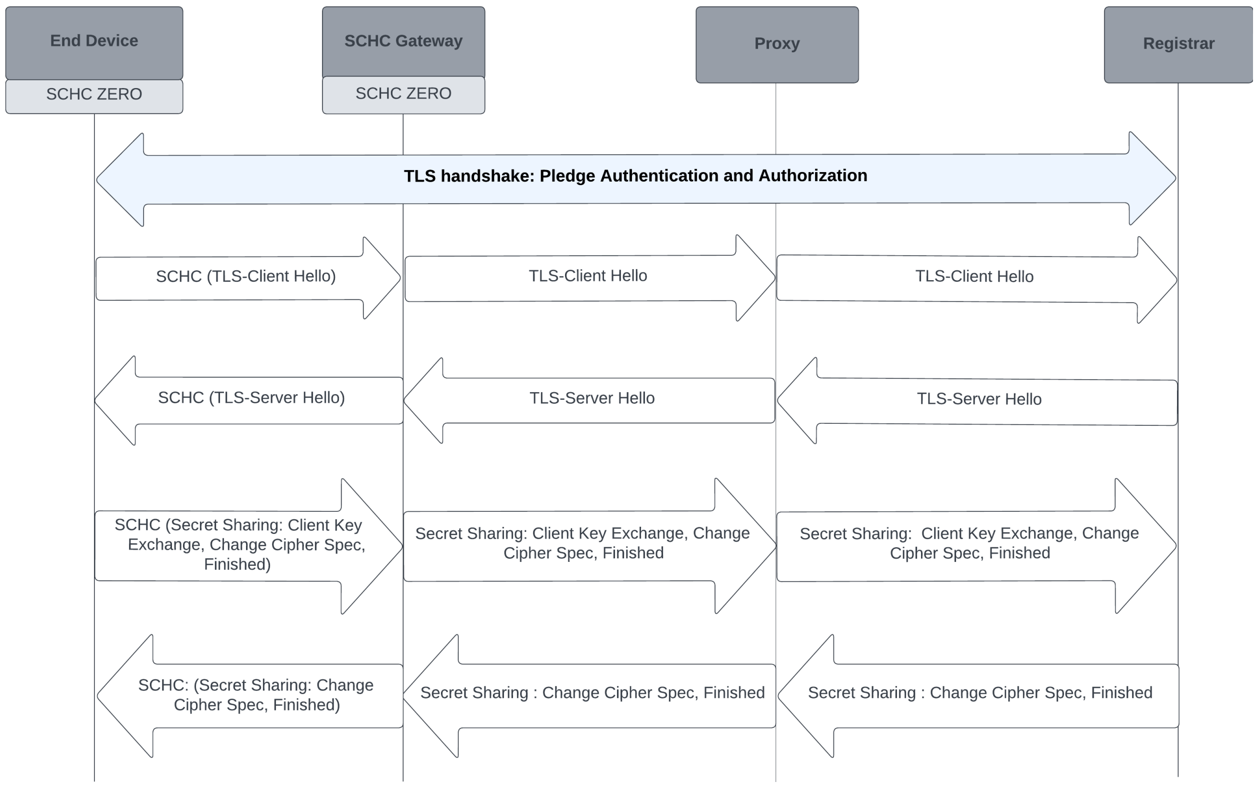

End Device
SCHC ZERO
SCHC Gateway
SCHC ZERO
Proxy
Registrar
TLS handshake: Pledge Authentication and Authorization
SCHC (TLS-Client Hello)
TLS-Client Hello
TLS-Client Hello
SCHC (TLS-Server Hello)
TLS-Server Hello
TLS-Server Hello
SCHC (Secret Sharing: Client Key Exchange, Change Cipher Spec, Finished)
Secret Sharing: Client Key Exchange, Change Cipher Spec, Finished
Secret Sharing: Client Key Exchange, Change Cipher Spec, Finished
SCHC: (Secret Sharing: Change Cipher Spec, Finished)
Secret Sharing : Change Cipher Spec, Finished
Secret Sharing : Change Cipher Spec, Finished

Figure 6.7: TLS/ DTLS Handshake Sequence Diagram

After the TLS handshake as indicated in the Figure 6.7, a secure tunnel is established first between the pledge and the Registrar, and subsequently between the Registrar and the MASA. This is crucial for the secure exchange of certificates and onboarding information.

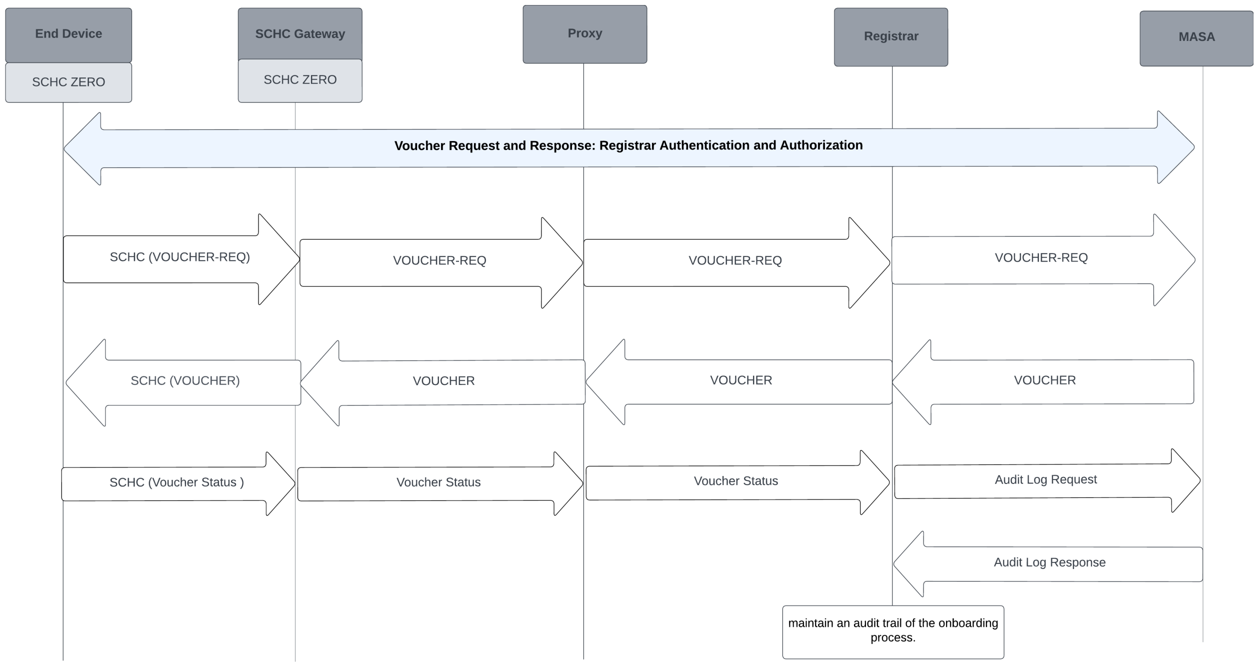


Figure 6.8: Voucher Request and Response Sequence Diagram

The Figure 6.8 shows the voucher on-boarding process. The sequence begins with the End Device initiating a voucher request (SCHC voucher request) that is propagated through the SCHC Gateway, Proxy, and Registrar until it reaches the MASA. The MASA then issues a voucher, which is sent back through the same path to the End Device. After the voucher is used, the End Device generates voucher status telemetry, which is also sent back through the same chain. Upon receiving the voucher status, the Registrar forwards it to the MASA, which then generates an audit log request and sends back an audit log response. This audit log helps maintain a secure and verifiable audit trail of the entire onboarding process, ensuring that the devices are correctly authenticated and authorized for network access.

## 6.3 ACPs to provide API for upper-level use

The final component involves using the Autonomic Control Plane (ACP) to provide an API for upper-level applications. This API allows for secure communication and management of devices within the network, building on the foundation laid by BRSKI.

The Figure 6.9 presents the system network stack design for establishing the management plane using the EST protocol, which is a critical step for initiating the data plane within a secure IP management framework. It illustrates the protocol layers across various entities, from the End Device to the CA, showing how BRSKI/cBRSKI protocols are employed for secure onboarding and management. The stack begins with LoRaWAN and SCHC at the End Device, transitioning through IPv6, TCP/UDP, and higher-level protocols like HTTP/COAP as the communication moves across the network infrastructure, including the Network Server, SCHC Gateway, Circuit Join Proxy, Registrar, and ultimately the CA. The diagram highlights the integration of different networking technologies, such as LoRaWAN and Ethernet, within the broader IPv6 ecosystem, emphasizing the secure exchange of data once the management plane is established.

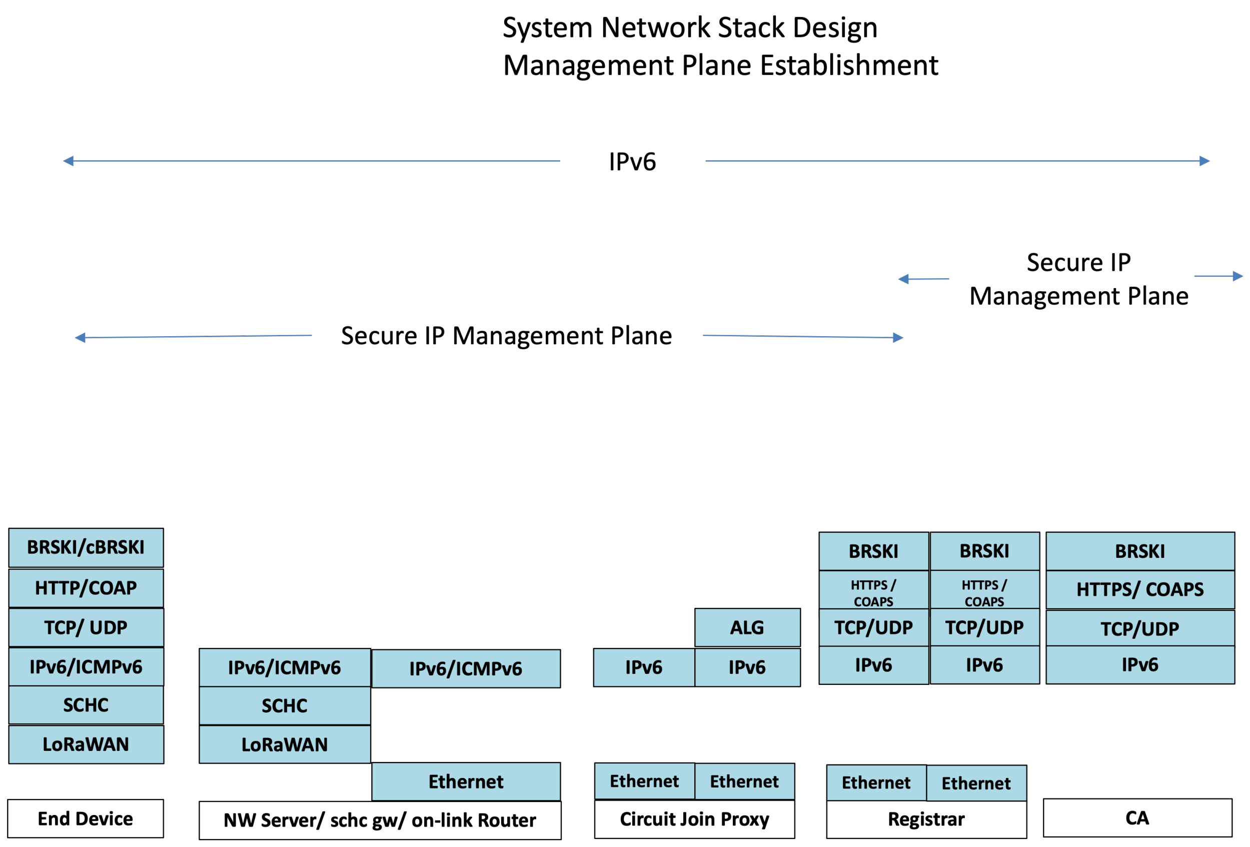


Figure 6.9: System Network Stack Design Management Plane Establishment in EST process

The Figure 6.10 presents a sequence diagram detailing the EST process in a BRSKI context. The diagram illustrates the communication between several entities: the End Device, SCHC Gateway, Proxy, Registrar, CA, and the IPv6 End Device. The process begins with the End Device initiating the EST enrollment by sending a combination of enrollment status, CSR attributes, and client certificate requests through the SCHC Gateway, Proxy, and Registrar, until it reaches the CA. The CA processes the request and returns the client certificate, which is then relayed back through the same chain to the End Device. Once the enrollment is complete, secure data exchange can occur, ensuring that the device is authenticated and securely integrated into the network. This diagram effectively visualizes the flow of EST messages and the involvement of different entities in securely enrolling a device in an IPv6 network using BRSKI.

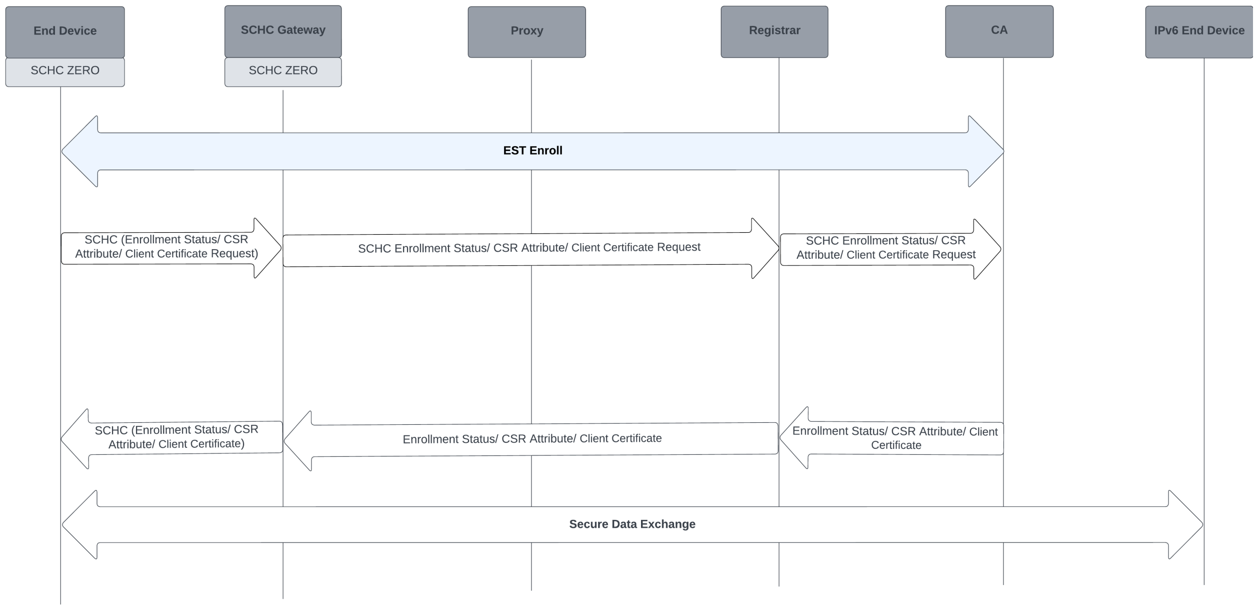

End Device
SCHC ZERO
SCHC Gateway
SCHC ZERO
Proxy
Registrar
CA
IPv6 End Device
EST Enroll
SCHC (Enrollment Status/ CSR Attribute/ Client Certificate Request)
SCHC Enrollment Status/ CSR Attribute/ Client Certificate Request
SCHC Enrollment Status/ CSR Attribute/ Client Certificate Request
SCHC (Enrollment Status/ CSR Attribute/ Client Certificate)
Enrollment Status/ CSR Attribute/ Client Certificate
Enrollment Status/ CSR Attribute/ Client Certificate
Secure Data Exchange

Figure 6.10: EST Sequence Diagram

With the ACP providing a secure API for upper-level applications, the system is now fully integrated and capable of supporting autonomic operations. This API facilitates communication, configuration, and management tasks, ensuring that the network can operate securely, in an autonomic way.

# Chapter 7

# Description of the Solution

Before initiating the autonomic onboarding process for devices, a set of predefined SCHC rules, which we called SCHC zero context, must be stored on both the end device and the SCHC gateway to enable transmission from the LoRaWAN to the IPv6 network. These rules handle the compression and fragmentation of large packets that exceed the channel's MTU size, which leads to an overhead to the transmission time.
In this section, we first present the SCHC packet transmission model. Next, we outline the SCHC Zero contexts necessary for the onboarding process. Finally, we calculate the Time on Air (ToA) for the packets in various scenarios by determining the packet sizes, including their header and payload sizes.

## 7.1 Message exchanges and fragmentation model

This section explains how messages are transmitted in the SCHC ACK-on-Error mode within a LoRaWAN network. Figure 7.1 illustrates the sequence of uplink and downlink messages using SCHC fragmentation. In this mode, each uplink fragment is sent after the device opens two designated time slots, known as receiving windows, which are indicated by the RD2 and RD1 intervals. Once all fragments for a given window are sent, the receiver sends an acknowledgment (ACK) to confirm successful transmission before the next set of fragments can be sent.

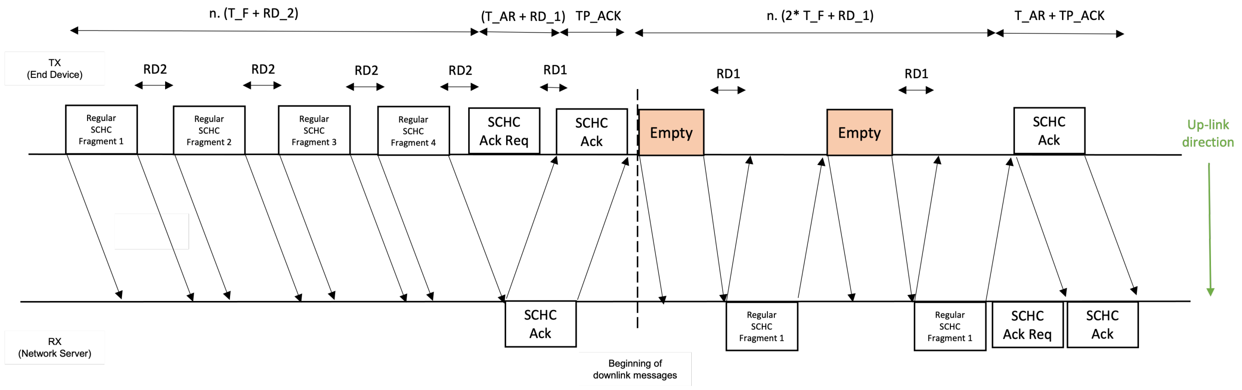


Figure 7.1: Up link and Down link messages in LoRaWAN using SCHC Fragmentation. Adapted from [2].

In the depicted scenario, it is assumed that there is no packet loss. Once the fragments for the current window are sent, the SCHC ACK is received. If there are more fragments to be sent for a message, after the

RD time, an empty packet is sent to the downlink. This action reopens the uplink window for the next set of fragments. This process continues until the entire SCHC message is successfully transferred. The figure also shows the sequence of sending regular SCHC fragments, receiving acknowledgments, and the use of empty packets to manage the flow control and ensure the successful delivery of all fragments.

## 7.2 SCHC Zero Context

Given that BRSKI facilitates the connection between constrained and non-constrained networks, it becomes essential to transmit messages with IPv6 headers in SCHC format. However, before starting BRSKI, no context can be exchanged between the two ends.

To tackle this challenge, we introduce the SCHC Zero context. This context established within the device before deployment time, eliminating the need for additional SCHC Context configuration. Once the bootstrap process ends, the new SCHC context for ongoing communication will be updated on the system in a secure way.

Table in 7.1 illustrates the messages within the system where header parameters should be stored as SCHC Zero context. All the fields indicated to be compressed by SCHC are the corresponding message header fields.

Table 7.1: On-boarding messages that require SCHC Zero context

| **Message** | **Headers** | **Fields to be compressed by SCHC Zero** |
|---|---|---|
| SLAAC | IPv6 header | IP Version, traffic flow label, Next Header, Hop limit, IPv6 app prefix, IPv6 DevIID, Destination address |
|  | ICMPv6 | ICMPv6 code , ICMPv6 type |
| M-Flood | IPv6 header | IP Version, traffic flow label, Next Header, Hop limit, IPv6 app prefix, IPv6 DevIID, Destination address |
|  | UDP | UDP Dev Port, UDP App Port, UDP Length, UDP Checksum |
| TLS Handshake | IPv6 header | IP Version, traffic flow label, Next Header, Hop limit, IPv6 app prefix, IPv6 DevIID, Destination address |
| DTLS Handshake/Voucher/EST | IPv6 header | IP Version, traffic flow label, Next Header, Hop limit, IPv6 app prefix, IPv6 DevIID, Destination address |
|  | UDP | UDP Dev Port, UDP App Port, UDP Length, UDP Checksum |

In the SLAAC process, the next header is ICMPv6, indicated by the value 58. Additionally, the DevIID is derived from the LoRa DevEUI 64 address, which is sent as the CDA field. The destination address and the IPv6 application prefix are the solicited-node multicast address and the link-local prefix, respectively, both of which are default values.

Table 7.2: SLAAC SCHC rules

| Field | Length (bits) | Target Value | Matching Operator | CDA |
|---|---|---|---|---|
| IPv6 version | 4 | 6 | equal | not sent |
| Traffic flow lable | 20 | (e.g., 0x00000) | match mapping | mapping-sent |
| Next header | 8 | 58 (ICMPv6) | equal | not sent |
| Hop limit | 8 | 1 | equal | not sent |
| IPv6 App prefix | 64 | FE80::/64 | equal | not sent |
| IPv6 DevIID | 64 | (derived from DevEUI 64) calculate in AS | ignore | DevIID |
| Destination address | 128 | FF02::1:FF00:0000/104 (Solicited-node multicast address ) | equal | not sent |

Table 7.3: SLAAC next header SCHC rules

| Field | Length (bits) | Target Value | Matching Operator | CDA |
|---|---|---|---|---|
| ICMPv6 type | 8 | 135 (NS) | equal | not sent |
| ICMPv6 code | 8 | 0 | equal | not sent |

According Tables 7.2 and 7.3 when applying SCHC to the SLAAC NS IPv6 and ICMPv6 contexts, several fields in the IPv6 and ICMPv6 headers are significantly reduced in size. For the IPv6 context, fields such as the IPv6 Version, Next Header, Hop Limit, IPv6 Application Prefix, and Destination Address are marked as Not Sent,leading to a reduction of 4 bits, 8 bits, 8 bits, 64 bits, and 128 bits, respectively. Although the Traffic Flow Label is compressed using a mapping mechanism, we assume no reduction for now due to the dependency on the mapping implementation. The IPv6 DevIID is sent without reduction. For the ICMPv6 context, both the ICMPv6 Type and Code fields, originally 8 bits each, are also not sent, resulting in an additional reduction of 16 bits. Summing up all these reductions, the total compressed size for the IPv6 context is reduced by 236 bits, and the ICMPv6 context by 16 bits, making the total reduction 252 bits or approximately 31.5 bytes.

Initially, the original size of the IPv6 packet was 40 bytes (320 bits). After applying SCHC compression, the compressed size is reduced to 68 bits.

Table 7.4: M-Flood IPv6 SCHC rules

| Field | Length (bits) | Target Value | Matching Operator | CDA |
|---|---|---|---|---|
| IPv6 version | 4 | 6 | equal | not sent |
| Traffic flow lable | 20 | (e.g., 0x00000) | match mapping | mapping-sent |
| Next header | 8 | 17 (UDP) | equal | not sent |
| Hop limit | 8 | 1 | equal | not sent |
| IPv6 App prefix | 64 | FE80::/64 | equal | not sent |
| IPv6 DevIID | 64 | (derived from DevEUI 64) calculate in AS | ignore | DevIID |
| Destination address | 128 | FF02::1:FF00:0000/104 (Solicited-node multicast address ) | equal | not sent |

Table 7.5: MFLOOD next header SCHC rules

| Field | Length (bits) | Target Value | Matching Operator | CDA |
|---|---|---|---|---|
| UDP dev port | 16 | 7017 (GRASP listen port) | equal | value sent |
| UDP app port | 16 | - | ignore | value sent |
| UDP length | 16 | - | ignore | compute-* |
| UDP checksum | 16 | - | ignore | compute-* |

Tables 7.4 and 7.5 show the contexts for the M-Flood IPv6 messages and its next header (UDP). The UDP TV code is 17, the UDP dev GRASP listen port is indicated by 7017 TV. The UDP length and checksum are computed once the packet is received to the destination using its payload information. Similar to the aforementioned tables, the total packet header size reduction amounts to 248 bits, compressing the original 48-byte (40 bytes for IPv6 and 8 bytes for UDP) headers to 17 bytes (136 bits).

Table 7.6: TLS IPv6 SCHC rules

| Field | Length (bits) | Target Value | Matching Operator | CDA |
|---|---|---|---|---|
| IPv6 version | 4 | 6 | equal | not sent |
| Traffic flow lable | 20 | (e.g., 0x00000) | match mapping | not sent |
| Next header | 8 | 6 (TCP) | equal | not sent |
| Hop limit | 8 | 1 | equal | not sent |
| IPv6 App prefix | 64 | FE80::/64 | equal | not sent |
| IPv6 DevIID | 64 | (derived from DevEUI 64) calculate in AS | ignore | DevIID |
| Destination address | 128 | - | ignore | sent |

Table 7.6 presents the contexts stored for TLS handshakes and message exchanges. Since the next header in TLS is TCP, the TV for this field is indicated by its code, which is 6. Unlike in SLAAC, the destination address is not a multicast address at this stage, as it has already been determined. The message header in this table is reduced by 112 bits through SCHC compression (IPv6 Version: 4 bits, Traffic Flow Label: 20 bits, Next Header: 8 bits, Hop Limit: 8 bits, IPv6 App Prefix: 64 bits, and Destination Address: 0 bits as it is sent).

Table 7.7: DTLS IPv6 SCHC rules

| Field | Length (bits) | Target Value | Matching Operator | CDA |
|---|---|---|---|---|
| IPv6 version | 4 | 6 | equal | not sent |
| Traffic flow lable | 20 | (e.g., 0x00000) | match mapping | not sent |
| Next header | 8 | 17 (UDP) | equal | not sent |
| Hop limit | 8 | 1 | equal | not sent |
| IPv6 App prefix | 64 | FE80::/64 | equal | not sent |
| IPv6 DevIID | 64 | (derived from DevEUI 64) calculate in AS | ignore | DevIID |
| Destination address | 128 | - | ignore | sent |

Table 7.8: DTLS next header SCHC rules

| Field | Length (bits) | Target Value | Matching Operator | CDA |
|---|---|---|---|---|
| UDP dev port | 16 | - | equal | value sent |
| UDP app port | 16 | 443(DTLS) | equal | not sent |
| UDP length | 16 | - | ignore | compute-* |
| UDP checksum | 16 | - | ignore | compute-* |

Tables 7.7 and 7.8 show the SCHC context zero for the DTLS messages and its next header (UDP) after MFlood. The 443 TV is set to define DTLS in the UDP app port. The compute-* field indicates that UDP length and UDP checksum can be calculated from the packet payload on the receiver side and it is not required to be transmitted. The message header in the table is reduced by 104 bits through SCHC compression (IPv6 Version: 4 bits, Traffic Flow Label: 20 bits, Next Header: 8 bits, Hop Limit: 8 bits, IPv6 App Prefix: 64 bits, and Destination Address: 0 bits as it is sent).

## 7.3 Calculating the on-boarding completion time

In the previous sections, we introduced the SCHC packet transmission model and the concept of SCHC Zero context, detailing how these mechanisms are used to compress and manage packet headers for efficient transmission over LoRaWAN. We also outlined the specific SCHC rules and contexts required for the on-boarding process, providing a comprehensive breakdown of how packet sizes are reduced.

Building on this foundation, this section will utilize the information from these earlier sections to calculate the ToA for various BRSKI and cBRSKI messages. This calculation is crucial for demonstrating that on-boarding completion time for the proposed system is reasonable across different LoRaWAN SFs in both BRSKI and cBRSKI scenarios. By systematically applying the SCHC Zero context and evaluating the resulting transmission times, we aim to prove that our approach ensures transmission time remains reasonable and efficient under all conditions, thereby validating the robustness of our system in constrained IoT environments.

Before proceeding to the packet sizes, it is important to understand the key differences between BRSKI and cBRSKI, which contribute to variations in payload sizes and header overheads. The Table 7.9 provides a comparison between BRSKI and cBRSKI:

Table 7.9: Comparison between BRSKI and cBRSKI

| **Feature** | **BRSKI** | **cBRSKI** |
|---|---|---|
| **Payload Encoding** | JSON format | CBOR format (reduces size) |
| **Signature Format** | CMS signature | COSE signature (encoded and secure, reduced in size) |
| **Transport Protocol** | HTTPS over TLS over TCP | COAPS over DTLS over UDP (less header overhead) |
| **EST** | EST | EST over COAPS (with shorter URIs, reduces message size) |

Now that we understand the differences between BRSKI and cBRSKI, particularly in terms of payload size and header overhead, we can proceed to calculate the ToA for the bootstrapping process.

The header size of each packet is predefined by its packet type and the next header field after being compressed by SCHC Zero. Also, we derived the packet sizes from the BRSKI source [22], and the provided project on their GitHub [47] to find the packet sizes after being compressed by their signatures. Below is Table 7.10 to show the packet sizes transferred during the on-boarding process. Note that the header sizes are the headers after SCHC Zero compression.

Table 7.10: BRSKI and cBRSKI Packet sizes

| Packet type | BRSKI Header size (B) | BRSKI Packet size (B) | cBRSKI Header size (B) | cBRSKI Payload size (B) |
|---|---|---|---|---|
| NS | 8 | 20 | 8 | 20 |
| M-Flood | 92 | 53 | 50 | 53 |
| TLS/DTLS client hello | 77 | 186 | 37 | 186 |
| TLS/DTLS server hello | 77 | 70 | 37 | 70 |
| TLS/DTLS secret sharing | 77 | 3061 | 37 | 3061 |
| TLS/DTLS change cipher | 77 | 149 | 37 | 149 |
| Voucher request | 77 | 1576 | 37 | 978 |
| Voucher Response | 77 | 1180 | 37 | 298 |
| Voucher status telemetry request | 77 | 114 | 37 | 114 |
| EST enrollment cert request | 77 | 678 | 37 | 678 |
| EST enrollment client certificate | 77 | 4246 | 37 | 4246 |

The reduced payload sizes in cBRSKI are attributed to the different signature methods employed, which decrease the payload size. Additionally, the smaller header size in cBRSKI compared to BRSKI is due to the use of the COAP header in cBRSKI. This COAP header can be compressed by SCHC, whereas the HTTP header used in BRSKI is not supported by SCHC for compression. Among the messages above, each of them that exceeds the MTU size of the LoRa device SF, will be fragmented in the SCHC adaptation layer.

### 7.3.1 Time on Air calculation

The Time on Air is a crucial parameter for calculating the time it it takes for the transmitter to insert a packet into the wireless medium using LoRa physical parameters, including the spreading factors. Figure 7.2 gives the pseudo-code of the ToA calculation :

```
Function CalculateTransmissionTime(SF, Payload_size, UpnLink):
    Set Tpr = (8 + 4.25) * (2^SF / 125)  // preamble transmission time
    Define fragmentation parameters: RD1, RX1, RD2, RX2, SCHC_header

    If SF is 12 to 7:
        Set AR and N based on SF

    Else:
        Print "Unsupported Spreading Factor"
        Return

    Initialize FCN, Tphysical, ToA_frf, ToA_prf, payloadSize_prf //rf: full
regular fragment, prf: regular partial fragment

    If Payload _size <= N - SCHC_header:
        Set payload = P
        Calculate Tphysical
        Set ToA = Tpr + Tphysical
        Return ToA as transmissionTime

    Else:
        Set tile_size = 10
        Calculate n_tiles_rf and n_T
        Initialize t_w, n_tiles_wn, N_w, RD_downlink

        For each window i from 0 to N_w - 1:
            If not the last window:
                Calculate mi and payloadSize_prf
            Else:
                Calculate mi and payloadSize_prf for last window

            Set payload = P
            Calculate Tphy and ToA_frf for full fragment
            Calculate Tphy and ToA_prf for partial fragment

            If UpnLink is 0:
                Calculate RD_downlink

            Update t_w with ToA_frf, ToA_prf, AR, and RD_downlink

        Return t_w as transmissionTime
```

Figure 7.2: on-boarding completion time pseudo-code

Figure 7.2 corresponds to the pseudo-code for CalculateTransmissionTime function is designed to compute the transmission time of messages sent to and from a LoRaWAN system. It handles both fragmented and non-fragmented messages, adjusting calculations based on the Spreading Factor (SF), payload size (P), and whether the message is uplink (Up-Link) or downlink.

**Initialization and Basic Calculation**
The function starts by calculating the preamble transmission time (Tpr) using the SF, which sets the foundation for further computations.

**Setting Fragmentation Parameters**
Constants such as RD1, RX1, RD2, RX2, and `SCHC_header` are defined, along with AR and N based on SF, to handle fragmentation.

**Handling Non-Fragmented Messages** If the payload size (`Payload_size`) is small enough, the function calculates the total transmission time (ToA) as the sum of Tpr and the physical transmission time (Tphy).

**Handling Fragmented Messages**
For larger payloads, the function calculates the number of tiles and windows needed for fragmentation. It then computes the transmission times for full regular fragments (ToA_frf) and regular partial fragments (ToA_prf), including any downlink delays. The total transmission time (t_w) is the sum of these.

**Components of Transmission Time**
Non-Fragmented Packets: Sum of Tpr and Tphy. Fragmented Packets: Includes times for full and partial fragments, additional delay in LoRa Class A, and any downlink delays. This ensures accurate calculation for various payload sizes and conditions in a LoRaWAN system.

To verify the accuracy of our outputs, we compared our ToA results with those generated by The Things Network's ToA calculator [48]. Using various packet sizes and device settings for the corresponding bandwidths, we found that our results were consistent with those provided by the website.

Finally, we used this function to calculate the ToA for all of the BRSKI and cBRSKi messsages. The results and analysis are provided in the next chapter.

# Chapter 8

# Results and Discussion

In the previous chapter, we detailed the SCHC packet transmission model and introduced the concept of SCHC Zero context, outlining how these mechanisms are leveraged to enable the autonomic on-boarding of LoRaWAN nodes into the IP network. We also calculated the on-boarding completion time for various BRSKI and cBRSKI messages, demonstrating the efficiency of our approach under different network conditions.

Building upon this foundation, this chapter presents the results derived from our implementation and analysis. We begin by examining the impact of SCHC Zero on the transmission times of BRSKI and cBRSKI messages across various spreading factors. Additionally, we explore the security synergy between LoRaWAN and BRSKI, providing a comprehensive overview of how these technologies work together to create a secure and autonomic management plane for the management of SCHC context updates. Finally, we discuss potential directions for future work, suggesting enhancements that could further improve the system's performance and security.

## 8.1 Defining SCHC Zero

A significant challenge arises when considering the connectivity of LoRaWAN end devices to IPv6 networks. Before establishing a connection, these devices require SCHC context sharing. Given that BRSKI facilitates the connection between constrained and non-constrained networks, it becomes essential to transmit messages with IPv6 headers in SCHC format. However, prior to securely bootstrapping the device into the network, no context can be exchanged between the two ends.

To address this challenge, we proposed a novel solution: SCHC zero context. This context is pre-configured within the device during the manufacturing process. Upon completion of the bootstrap process, the SCHC context required for further communication will be securely shared. The intricacies of this system have been elaborated upon in previous chapters, particularly in the section discussing the implementation and storage of SCHC zero contexts for various message types in the system.

The reduction in payload sizes is attributed to the different signature methods employed, which decrease the payload size. Additionally, the smaller header size in cBRSKI compared to BRSKI is due to the use of the COAP header in cBRSKI, which can be compressed by SCHC, whereas the HTTP header used in BRSKI is not supported by SCHC for compression.

## 8.2 Evaluating the on-boarding completion time

First, we checked the details of the messages passed in the system by checking the related resources [1,22,49]. We created a sequence diagram from the messages inclding their protocol stack and all the headers.

We implemented the time calculation using MATLAB by utilizing the LoRaWAN settings inputs from the LoRa Alliance website [49]. To ensure the accuracy of our implementation, we compared the time results of our code for non-fragmented data with the values provided by The Things Network website. This comparison helped verify the precision of our ToA calculations.

Additionally, we tested the MATLAB code's accuracy by creating diagrams that compare data size and SF with ToA. The relation between the payload size and different LoRa SFs are validated in Figure 8.1.

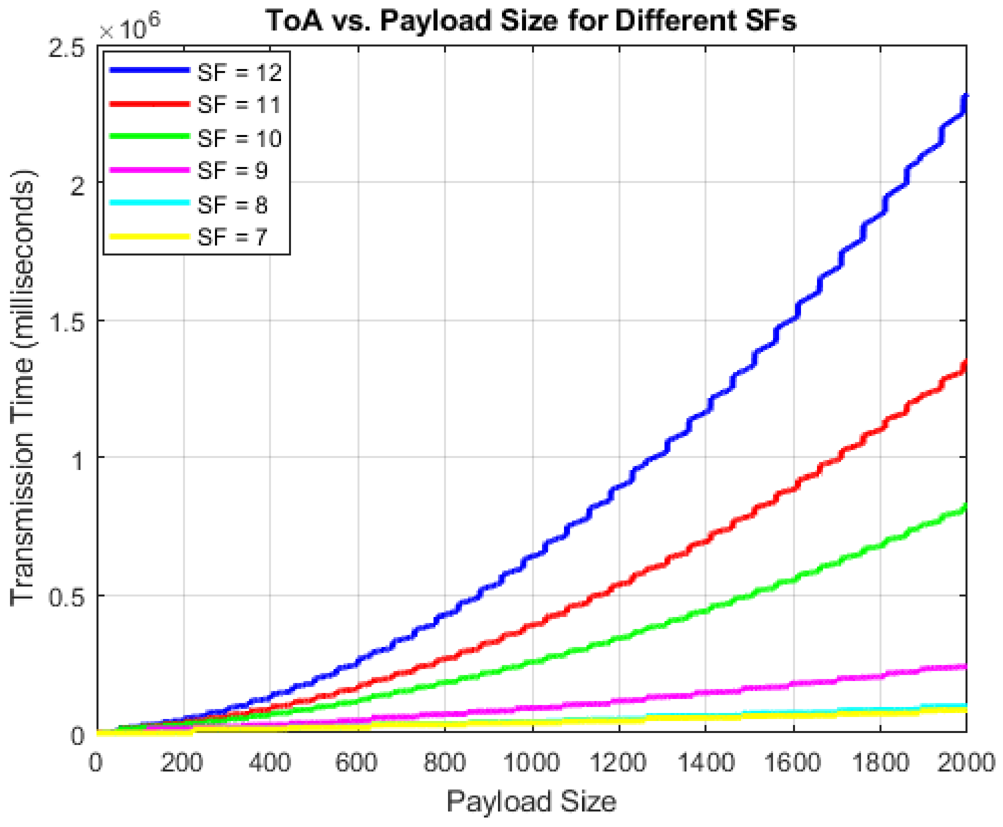


Figure 8.1: Transmission time for different payload size in each SF

The results demonstrate a clear exponential relationship between transmission time and payload size in LoRaWAN communications. As the payload size grows, the transmission time increases at an accelerating rate. This is because larger payloads require more time to be transmitted, and the additional data adds a proportionally larger amount of time to the overall transmission.

This exponential increase is further exacerbated by higher SFs. The Spreading Factor is a crucial parameter in LoRaWAN that affects the range, the data rate of the transmission, and the MTU. Higher SFs result in longer transmission times due to the lower data rate. When the SF increases, each bit of data takes longer to transmit, which means that for the same increase in payload size, the increase in transmission time is more significant at higher SFs compared to lower SFs.

After validating the code's functionality, we extracted message sizes from the BRSKI RFC and cBRSKI draft. We then implemented the BRSKI code provided by the authors on GitHub, taking into account the stored context zero compressed by SCHC.

Finally, we calculated the overall on-boarding completion time and compared the transmission times of cBRSKI with BRSKI using a LoRa Class A device. The details of these comparisons will be discussed in the next section.

The comparison of completion times between BRSKI and constrained BRSKI (cBRSKI) across various Spreading Factors (SFs) reveals significant insights into the efficiency of these protocols. The results demonstrate that cBRSKI consistently outperforms BRSKI in terms of transmission time, primarily due to the use of more efficient header formats (Figure 8.2). However, the gain is not siginifactive and could be reduced further with additional data compression.

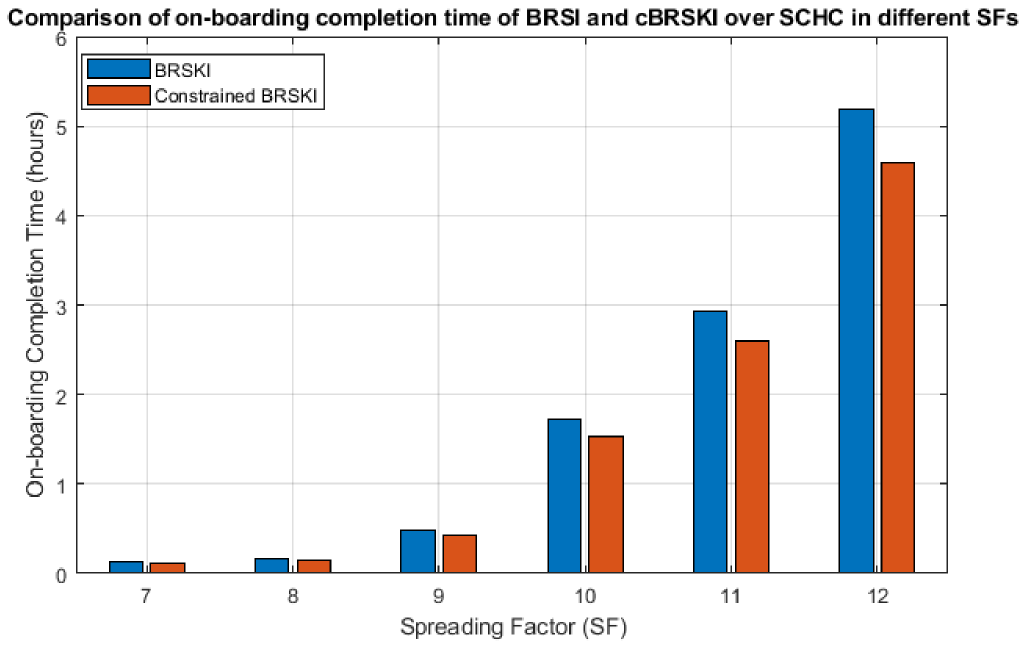


Figure 8.2: Comparison of the on-boarding completion time between BRSKI and cBRSKI with different LoRa SFs

For SF7 to SF12, the transmission time increases exponentially for both protocols. However, the increase is significantly steeper for BRSKI than for cBRSKI. This substantial reduction in transmission time for cBRSKI can lead to improved battery life and more efficient use of network resources, as devices spend less time transmitting data. The numerical results are indicated in the Table 8.1.

Table 8.1: Comparison of transmission time between BRSKI and cBRSKI (the numerical results)

| **SF** | **BRSKI (hours)** | **cBRSKI (hours)** |
|---|---|---|
| **7** | 0.13 | 0.11 |
| **8** | 0.16 | 0.14 |
| **9** | 0.48 | 0.42 |
| **10** | 1.73 | 1.53 |
| **11** | 2.94 | 2.60 |
| **12** | 5.18 | 4.59 |

The maximum transmission time for BRSKI is approximately 18,650 seconds, while for cBRSKI, it is approximately 16,527 seconds, corresponding to around 5.1 hours and 4.5 hours, respectively. Given that the onboarding process occurs only infrequently, these transmission times are considered both reasonable and effective for ensuring secure and efficient network integration.

cBRSKI uses COAP instead of HTTP and DTLS instead of TLS, as well as CBOR compression mechanism for the exchanged vouchers which results in reduced packet sizes and more efficient use of bandwidth. This is reflected in the lower transmission times for cBRSKI.
The reduced transmission times for cBRSKI are particularly beneficial in scenarios with high SFs, where the impact of longer transmission times is more pronounced.

## 8.3 Security Synergy Between LoRaWAN and BRSKI

In this section, we explore how the security mechanisms of LoRaWAN and Bootstrapping Remote Secure Key Infrastructure (BRSKI) work in tandem to enhance the overall security framework for IoT networks, particularly focusing on the link layer, IPv6 application layer and network layers. The table below summarizes the security aspects provided by each method.

Table 8.2: Security Metrics

| **Aspect** | **LoRaWAN** | **BRSKI** |
|---|---|---|
| Integrity and Authentication | As LoRaWAN OTAA operates on data link layer, it provides CMAC to ensure integrity and CTR for encryption using shared keys for authentication [21] | BRSKI operates at the application layer using built-in mechanisms for ensuring data integrity and reliability with UDP and TCP checksums |
| DoS Attack Prevention | LoRaWAN devices can use different channels and SFs for communication, making it harder for DoS [7] | Protects against MitM and bit flipping attacks by ensuring integrity and authenticity of exchanged keys/messages using digital signatures |
| Man in the Middle Attack | MICs are computed using the NwkSKey to ensure that messages have not been tampered | The pledge presents its identity certificate signed by MASA as a third party |
| Replay Attack Prevention | Frame counters that must be incremented with each message. Fresh session keys generated during the OTAA process to prevent reuse of old keys. | Unique nonce values and timestamps to prevent old messages being reused |

The table 8.2 shows how LoRaWAN ensures the integrity, confidentiality, and authenticity of data transmitted over low-power wide-area networks through a multi-layered security approach in the OSI data link layer. The key components include the AES cryptographic primitive combined with several modes of operation: CMAC2 for integrity protection and CTR3 for encryption. The session keys, specifically AppSKey and NwkSKey. The AppSKey encrypts application payloads end-to-end between the device and the application server, while the NwkSKey secures MAC commands and network communication.

Mutual authentication is achieved through a secure join procedure verified by a Join Server, ensuring that only authenticated devices can join the network. Additionally, Message Integrity Codes (MICs) are used to provide integrity checks and origin authentication, and frame counters are employed to prevent replay attacks by ensuring each message is unique.

Also, the comparison table shows that BRSKI provides a robust security framework for bootstrapping devices in the OSI upper layers including network layer and IPv6 application layers with a focus on authentication, authorization, and secure communication. It employs nonce-based replay protection to ensure the freshness and authenticity of messages, and encrypts communication to maintain data confidentiality. Certificates play a crucial role in BRSKI, with the MASA validating device authenticity and authorizing network access, ensuring that only legitimate devices can join the network.

As a result, LoRaWAN and BRSKI offer complementary security features that together create a robust security architecture for IoT networks. Enhanced replay protection is achieved through the use of frame counters in LoRaWAN and nonce-based mechanisms in BRSKI. Comprehensive encryption is provided by LoRaWAN for end-to-end LoRaWAN application payloads and by BRSKI for initial bootstrapping communication. Mutual authentication in LoRaWAN, which ensures that only authenticated devices join the network, is enhanced by BRSKI's certificate-based authentication and authorization, validated by MASA. By combining LoRaWAN's security measures at the MAC and LoRa application layers with BRSKI's security for bootstrapping, and IPv6 application network layers, a comprehensive, multi-layered security framework is established, addressing security concerns from device initialization to data transmission.

# Chapter 9

# Conclusion

This thesis explored the integration of LoRaWAN with IPv6 networks, focusing on enhancing the efficiency, security, and overall performance of data transmission in constrained environments. By addressing the challenges of device onboarding and communication between LoRaWAN and IPv6, we proposed and implemented a series of solutions that contribute to the broader field of IoT networking. The key contributions of this work are summarized as follows:

## 9.1 Key Contributions

- Introduction of SCHC Zero Context:

  One of the primary challenges in enabling seamless communication between LoRaWAN and IPv6 networks is the lack of shared SCHC context at the initial onboarding stage. To overcome this, we introduced the concept of SCHC Zero context, a pre-configured set of SCHC rules embedded within the device during manufacturing. This innovation allows devices to begin communicating immediately upon connection to the network, facilitating a secure and efficient bootstrapping process. The SCHC Zero context reduces the need for additional configuration, ensuring that the devices can be onboarded to the IPv6 network with minimal overhead. Evaluation of Transmission Time (ToA):

  We conducted an in-depth analysis of the Time on Air (ToA) for various BRSKI and cBRSKI messages, demonstrating that our approach ensures that the transmission time remains reasonable and efficient across different LoRaWAN Spreading Factors (SFs). By implementing a robust ToA calculation model, we validated that the proposed system performs effectively under all conditions, providing a practical solution for IoT applications that require reliable and timely data transmission.

- Comparative Analysis of BRSKI and cBRSKI:

  Through the implementation and comparison of BRSKI and cBRSKI protocols, we highlighted the advantages of using cBRSKI in terms of transmission efficiency. The results showed that cBRSKI outperforms BRSKI due to its use of more efficient header formats, such as COAP over UDP and CBOR for message compression. Moreover, the transmission time of maximum 5 hours for the on-boarding process, is effective which is crucial for energy-constrained IoT devices, as it directly impacts battery life and network resource utilization.

- Security Synergy Between LoRaWAN and BRSKI:

  The integration of LoRaWAN and BRSKI security mechanisms provides a comprehensive multi-layered security framework for IoT networks. By combining the data link layer security provided by LoRaWAN with the application and network layer security of BRSKI, we achieved enhanced protection against common threats such as replay attacks, man-in-the-middle (MitM) attacks, and denial of service (DoS) attacks. The synergy between these two protocols ensures that IoT devices are not only securely onboarded but also maintain secure communication throughout their operation.

## 9.2 Future Work

In addition to the current implementation, Future Work will explore further optimizations and security enhancements. Potential areas for development include the integration of Object Security for Constrained RESTful Environments (OSCORE) with SCHC [38], and the application of Diet-ESP [50] for further compression while maintaining security. These enhancements could provide even greater efficiency and security for IoT devices operating in constrained environments.

# Bibliography

[1] M. Pritikin, M. Richardson, T. Eckert, M. Behringer, and K. Watsen, “Bootstrapping remote secure key infrastructures (brski),” *IP Journal*, 2021.

[2] R. Muñoz, “Modeling and efficiency evaluation of the schc standard for ip packet transport over lorawan,” Master’s thesis, University of Chile, Faculty of Physical and Mathematical Sciences, Department of Electrical Engineering, 2020.

[3] T. Kaukalias and P. Chatzimisios, “Internet of things (iot),” in *Encyclopedia of Information Science and Technology, Third Edition*. IGI Global, 2015, pp. 7623–7632.

[4] S. H. Shah and I. Yaqoob, “A survey: Internet of things (iot) technologies, applications and challenges,” *2016 IEEE Smart Energy Grid Engineering (SEGE)*, pp. 381–385, 2016.

[5] R. Dillet. (2022) Sigfox, the french iot startup that had raised more than $300m, files for bankruptcy protection as it seeks a buyer. [Online]. Available: https://techcrunch.com/2022/01/27/sigfox-the-french-iot-startup-that-had-raised-more-than-300m-files-for-bankruptcy-protection-as-it-seeks-a-buyer

[6] K. Banti et al., “Lorawan communication protocols: A comprehensive survey under an energy efficiency perspective,” *Telecom*, vol. 3, no. 2, pp. 322–357, 2022.

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

[8] A. Feijoo-Añazco, D. Garcia-Carrillo, J. Sanchez-Gomez, and R. Marin-Perez, “Innovative security and compression for constrained iot networks,” *Internet of Things*, vol. 24, p. 100899, 2023.

[9] Sierra Wireless, “Sierra Wireless Official Website,” 2024, accessed: 2024-05-28. [Online]. Available: https://www.sierrawireless.com/?utm_term=semtech&utm_campaign=Brand-NAM-EN-LG&utm_source=google&utm_medium=cpc&hsa_acc=5534429107&hsa_cam=960596284&hsa_grp=121643025295&hsa_ad=692961838768&hsa_src=g&hsa_tgt=kwd-488284840512&hsa_kw=semtech&hsa_mt=e&hsa_net=adwords&hsa_ver=3&gad_source=1&gclid=EAIaIQobChMIqsaJs5iwhgMVAUlHAR1YXw8YEAAYASAAEgKssPD_BwE

[10] R. Muñoz, J. Saez Hidalgo, F. Canales, D. Dujovne, and S. Céspedes, “Schc over lorawan efficiency: Evaluation and experimental performance of packet fragmentation,” *Sensors*, vol. 22, no. 4, p. 1531, 2022.

[11] T. T. N. Foundation, “the things network lora classes,” The Things Network Industries, accessed: 2024, https://www.thethingsnetwork.org/docs/lorawan/classes/.

[12] B. Carpenter, “Grasp an introduction,” pp. 2–3, 2021, https://github.com/becarpenter/graspy/blob/master/documentation/GRASP-intro.pdf.

[13] C. Bormann, B. E. Carpenter, and B. Liu, “GeneRic Autonomic Signaling Protocol (GRASP),” RFC 8990, May 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc8990

[14] M. Pritikin, P. E. Yee, and D. Harkins, “Enrollment over Secure Transport,” RFC 7030, Oct. 2013. [Online]. Available: https://www.rfc-editor.org/info/rfc7030

[15] O. Gimenez and I. Petrov, “Static Context Header Compression and Fragmentation (SCHC) over LoRaWAN,” RFC 9011, Apr. 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc9011

[16] C. Gomez, A. Minaburo, L. Toutain, D. Barthel, and J. C. Zuniga, “Ipv6 over lpwans: Connecting low power wide area networks to the internet (of things),” *IEEE Wireless Communications*, vol. 27, no. 1, pp. 206–213, 2020.

[17] B. Moons, E. De Poorter, and J. Hoebeke, “Device discovery and context registration in static context header compression networks,” *Information*, vol. 12, no. 2, p. 83, 2021.

[18] E. Sisinni, D. Fernandes Carvalho, A. Depari, P. Bellagente, A. Flammini, M. Pasetti, S. Rinaldi, and P. Ferrari, “Assessing a methodology for evaluating the latency of ipv6 with schc compression in lorawan deployments,” *Sensors*, vol. 23, no. 5, p. 2407, 2023.

[19] A. Minaburo, L. Toutain, C. Gomez, D. Barthel, and J.-C. Zúñiga, “SCHC: Generic Framework for Static Context Header Compression and Fragmentation,” RFC 8724, Apr. 2020. [Online]. Available: https://www.rfc-editor.org/info/rfc8724

[20] T. T. N. Foundation, “the things network Lora device activation,” The Things Network Industries, accessed: 2024, https://www.thethingsnetwork.org/docs/lorawan/end-device-activation/.

[21] A. GEMALTO and SEMTECH, “LoRa Alliance Security,” Internet Engineering Task Force (IETF), 2017, https://lora-alliance.org/wp-content/uploads/2020/11/lorawan_security_whitepaper.pdf.

[22] M. Richardson, P. V. der Stok, P. Kampanakis, and E. Dijk, “Constrained Bootstrapping Remote Secure Key Infrastructure (cBRSKI),” Internet Engineering Task Force, Internet-Draft draft-ietf-anima-constrained-voucher-25, Jul. 2024, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-anima-constrained-voucher/25/

[23] W. Ayoub, M. Mroue, F. Nouvel, A. E. Samhat, and J.-C. Prévotet, “Towards ip over lpwans technologies: Lorawan, dash7, nb-iot,” in *2018 sixth international conference on digital information, networking, and wireless communications (dinwc)*. IEEE, 2018, pp. 43–47.

[24] S. Aguilar et al., “Packet fragmentation over sigfox: Implementation and performance evaluation of schc ack-on-error,” *IEEE Internet of Things Journal*, vol. 9, no. 13, pp. 11 057–11 070, 2021.

[25] S. Aguilar et al, “Lorawan schc fragmentation demystified,” *International Conference on Ad-Hoc Networks and Wireless*, 2019.

[26] S. Aguilar et al., “Performance analysis and optimal tuning of ietf lpwan schc ack-on-error mode,” *IEEE Sensors Journal*, vol. 20, no. 23, pp. 14 534–14 547, 2020.

[27] S. Aguilar, R. Vidal, and C. Gomez, “Evaluation of receiver-feedback techniques for fragmentation over lpwans,” *IEEE Internet of Things Journal*, vol. 9, no. 9, pp. 6866–6878, 2021.

[28] D. S. Wistuba La-Torre, S. Céspedes, and J. Bustos-Jiménez, “Modeling schc ack-on-error fragment delivery over sigfox,” in *Proceedings of the 18th ACM International Symposium on QoS and Security for Wireless and Mobile Networks*, 2022, pp. 115–119.

[29] S. Aguilar, A. Platis, R. Vidal, and C. Gomez, "Energy consumption model of schc packet fragmentation over sigfox lpwan," *Sensors*, vol. 22, no. 6, p. 2120, 2022.

[30] K. Q. Abdelfadeel, V. Cionca, and D. Pesch, "Lschc: Layered static context header compression for lpwans," in *Proceedings of the 12th Workshop on Challenged Networks*, 2017, pp. 13–18.

[31] D. Wistuba et al., "An implementation of iot lpwan schc message fragmentation and reassembly," in *SSN 2020: V School on Systems and Networks: proceedings of the V School on Systems and Networks, SSN 2020: Vitória, Brazil, December 14-15, 2020.* CEUR-WS. org, 2020, pp. 1–4.

[32] J. Sanchez-Gomez, J. Gallego-Madrid, R. Sanchez-Iborra, J. Santa, and A. F. Skarmeta, "Impact of schc compression and fragmentation in lpwan: A case study with lorawan," *Sensors*, vol. 20, no. 1, p. 280, 2020.

[33] J. Haxhibeqiri, E. De Poorter, I. Moerman, and J. Hoebeke, "A survey of lorawan for iot: From technology to application," *Sensors*, vol. 18, no. 11, p. 3995, 2018.

[34] M. A. Ertürk, M. A. Aydın, M. T. Büyükakkaşlar, and H. Evirgen, "A survey on lorawan architecture, protocol and technologies," *Future internet*, vol. 11, no. 10, p. 216, 2019.

[35] R. Sanchez-Iborra, J. Sánchez-Gómez, S. Pérez, P. J. Fernández, J. Santa, J. L. Hernández-Ramos, and A. F. Skarmeta, "Internet access for lorawan devices considering security issues," in *2018 Global Internet of Things Summit (GIoTS).* IEEE, 2018, pp. 1–6.

[36] G. Álvarez, J. A. Fraire, K. A. Hassan, S. Céspedes, and D. Pesch, "Uplink transmission policies for lora-based direct-to-satellite iot," *IEEE Access*, vol. 10, pp. 72 687–72 701, 2022.

[37] S. T. Arzo, C. Naiga, F. Granelli, R. Bassoli, M. Devetsikiotis, and F. H. Fitzek, "A theoretical discussion and survey of network automation for iot: Challenges and opportunity," *IEEE Internet of Things Journal*, vol. 8, no. 15, pp. 12 021–12 045, 2021.

[38] L. E. V. Zavala, A. O. García, and M. Siller, "Architecture and algorithm for iot autonomic network management," in *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData).* IEEE, 2019, pp. 861–867.

[39] R. F. Sari, L. Rosyidi, B. Susilo, and M. Asvial, "A comprehensive review on network protocol design for autonomic internet of things," *Information*, vol. 12, no. 8, p. 292, 2021.

[40] M. Tahir, Q. M. Ashraf, and M. Dabbagh, "Towards enabling autonomic computing in iot ecosystem," in *2019 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech).* IEEE, 2019, pp. 646–651.

[41] S. Afzal, L. C. De Biase, G. Fedrecheski, W. T. Pereira, and M. K. Zuffo, "Analysis of web-based iot through heterogeneous networks: Swarm computing over lorawan," *Sensors*, vol. 22, no. 2, p. 664, 2022.

[42] B. Moons, A. Karaagac, J. Haxhibeqiri, E. De Poorter, and J. Hoebeke, "Using schc for an optimized protocol stack in multimodal lpwan solutions," in *2019 IEEE 5th World Forum on Internet of Things (WF-IoT).* IEEE, 2019, pp. 430–435.

[43] K. Q. Abdelfadeel, V. Cionca, and D. Pesch, “Dynamic context for static context header compression in lpwans,” in *2018 14th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE, 2018, pp. 35–42.

[44] W. Ayoub, A. E. Samhat, F. Nouvel, M. Mroue, H. Jradi, and J.-C. Prévotet, “Media independent solution for mobility management in heterogeneous lpwan technologies,” *Computer Networks*, vol. 182, p. 107423, 2020.

[45] A. Bernard, S. Balakrichenan, M. Marot, and B. Ampeau, “Dns-based dynamic context resolution for schc,” in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

[46] M. Holdrege and P. Srisuresh, “IP Network Address Translator (NAT) Terminology and Considerations,” RFC 2663, Aug. 1999. [Online]. Available: https://www.rfc-editor.org/info/rfc2663

[47] M. B. Brian Carpenter, Bing Liu, “ANIMAGUS GitHub Repository, a reference implementation of RFC 8995,” 2024, https://github.com/ANIMAgus-minerva.

[48] T. T. N. Foundation, “The things network toa calculator,” The Things Network Industries, accessed: 2024, https://www.thethingsnetwork.org/airtime-calculator.

[49] D. Kjendal, “LoRa Regional Parameters,” LoRa Alliance, 2022, https://resources.lora-alliance.org/technical-specifications/rp002-1-0-4-regional-parameters.

[50] D. Migault et al., “ESP Header Compression Profile,” Internet Engineering Task Force, Internet-Draft draft-ietf-ipsecme-diet-esp-01, Jul. 2024, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-ipsecme-diet-esp/01/