ECDH Based Key Management for LoRaWAN Considering Sensor Node Limitations.

A dissertation submitted for the Degree of Master of Science in Computer Science

E. N. Jayasuriya
University of Colombo School of Computing
2019

Declaration

The thesis is my original work and has not been submitted previously for a degree at this or any other university/institute.
To the best of my knowledge it does not contain any material published or written by another person, except as acknowledged in the text.

Student Name: E. N. Jayasuriya
Registration Number: 2016/MCS/045
Index Number: 16440459

Signature: Date: 20.07.2019

This is to certify that this thesis is based on the work of
Mr. Eranda Namal Jayasuriya
under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Certified by:
Supervisor Name:

Signature: Date:

Abstract

LoRaWAN is Low Power Wide Area Network which uses LoRa radio technology as its sensor node level communication. LPWANs select LoRa as radio technology because of its long-distance communication ability with lower power consumption. Sensor networks have sensitive data and they should transfer to Application servers securely. Cryptographic algorithms need more computational power and memory space for their cryptographic calculations. These battery powered sensor devices have limited memory, limited processing power, limited battery power, and lower bandwidth with LoRa. Hence, all these limitations should consider while implementing security solutions. LoRaWAN has already implemented a security mechanism to send sensor data securely to end points, but still it has some security vulnerabilities. Literature review shows the need for a new key distribution and management mechanism for the LoRaWAN system. LoRa Alliance and other researchers have suggested improvements to the LoRaWAN existing joining mechanism and root key based key distribution mechanism. This research identifies the capability of running Elliptic Curve Diffie Hellman based key distribution and management mechanism, considering the limitations of the LoRaWAN End Nodes. Capability of running an existing implementation of ECDH has shown using the literature and experiments that have done in this work. The calculated power consumption percentage has shown that the power overhead for key distribution and management is bearable with the proposed mechanism by this research.

Acknowledgment

First of all, I would like to pay my gratitude to my supervisor, Dr. Kasun de Zoysa for his uninterrupted support and guidance throughout the master's program. I want to acknowledge the invaluable insights, comments, and support given by my co-supervisor Dr. Kasun Gunawardana.
Then I thank Research Project coordinators Dr. Lasanthi De Silva and Mrs. Kokila Kasuni for their guidance and keep us on the research track.
I would like to thank Mrs. Harshi Abeyrathne, Mr. Oshan Chinthaka and my dear friends working at the University of Colombo School of Computing for their great help. Next, want to thank my friend Mr. Dinith Minura and the other colleagues who did the Masters at University of Colombo School of Computing for helping me in many ways throughout the year of research. I also thank Prof. Ng Wee Keong for releasing me for my research work and Mr. Anupa Shamlal for his guidance and help.
Finally, I would like to thank my parents and my family members for their support for me to complete this masters program.

Table of Contents

List of Figures

List of Tables

List of Abbreviation

LoRa	: Long Range
LoRaWAN	: LoRa Wide Area Network
LPWAN	: Low Power Wide Area Network
ABP	: Activation by Personalise
OTAA	: Over the Air Activation
SF	: Spread Factor
DH	: Diffie Hellman
ECC	: Elliptic Curve Cryptography
ECDH	: Elliptic Curve Diffie Hellman
ECDSA	: Elliptic Curve Digital Signature Algorithm
MAC	: Message Authentication Code
MIC	: Message Integrity Code
EUI	: Extended Unique Identifier
AES	: Advanced Encryption Standards
MCU	: Micro Controller Unit
ACK	: Acknowledgement
IDE	: Integrated Development Environment
GPIO	: General Purpose Input Output
RAM	: Random Access Memory
EEPROM	: Electronically Erasable Programmable Memory
SRAM	: Static Random Access Memory
SSL	: Secure Socket Layer
TLS	: Transport Layer Security
HSM	: Hardware Secure Module
FLOPS	: Floating Point Operations
RSSI	: Received Signal Strength Indicator
OS	: Operating System

Chapter 1

Introduction

Data is emerging as the most valuable asset in the world. All the key domains including Economy, Education, Health and Business have coupled and are evolving with Information technology. Data is the molecule of Information Technology and securing data has become a necessity. This is an era which is named as *Era of Internet of Things* because all the digital media are getting connected to the internet. Going beyond the human day today using mobile devices like mobile phones and laptops, sensor nodes which may far away from the people are also connected to the Internet. A major portion of cloud data has become sensor data. The portion of those sensor nodes are battery powered, should last longer and rarely accessed by people. To reduce power consumption, these sensor nodes use Low Power Wide Area Networks to connect to the internet. These sensor nodes are equipped with microcontrollers which have limited processing power to make them less power consumable. Hence such a sensor system has hundreds to a thousand of sensors they should have less maintenance and should be remotely maintainable. When such a sensor system has sensitive data, it should use security mechanisms. These security mechanisms are operated on security keys and to manage security keys with less maintenance it needs a robust security key management protocol. This Research is focusing on how we can distribute and manage the security keys to ensure the security aspects of widely spread Low Power Wide Area Network called LoRaWAN considering the limitations of its Sensor Nodes.

1.1 Background

At the beginning of the Internet of things era, only a few peoples have the devices which are connected to the internet. Then with the technology goes up, an average every people owns a device which is connected to the internet. It means the number of devices connected to the internet is equal to the world population. Then with the technologies grows up each people have more than one day today using IOT devices such as mobile phone, laptop, desktop computer or tablet. Now

the world is becoming smart, smart solutions are introducing for each and every field using the internet of things. Now homes are becoming smart with electronic devices are controllable control through the internet. The vehicle is also connected to IOT for various smart solutions like traffic management, tracking, and real-time vehicle condition monitoring. Other than homes and vehicles other fields also becoming smart. As examples, Smart grids, smart farms, weather monitoring, disaster management systems and over the air parcel delivery can be considered. So in 2020, the number of IOT devices owned by a person is expected as six and totally more than fifty billion devices are expected. **Figure 1** shows the statistics of the usage of IOT throughout the past years and also future expectation over the population.

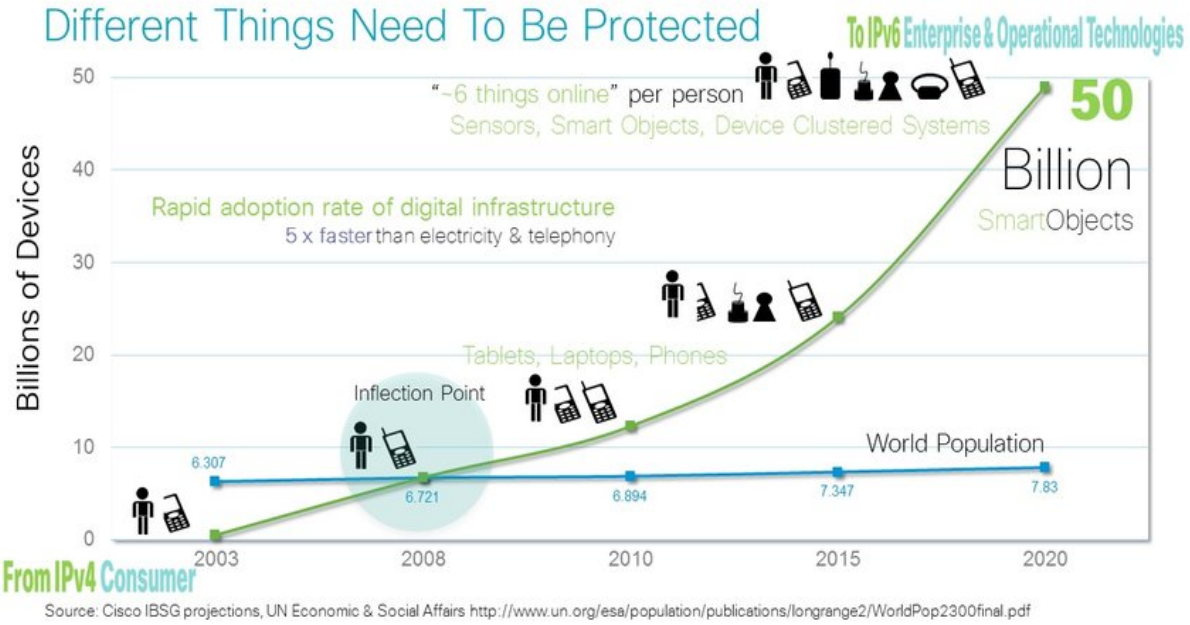


Figure 1: IOT devices usage and future expectation

When considering the Sensor systems, most sensor nodes are not generating data streams with high data rates. A sensor reading is typically less than 10 bytes and a data packet is less than 50 bytes. As an example, a data packet of a tracking device is shown in **Figure 2**. Most sensor nodes of sensor systems are battery powered because they are deployed far away areas where it difficult to supply grid power for each and every node. Sensor nodes are typically automotive and can be operated remotely. Maintaining them frequently by physically accessing is a huge problem because such a system has hundreds to thousands of nodes. So, if a node cannot survive for a long time period with a battery, maintenance becomes a huge problem. Examples for such sensor systems are weather monitoring sensor systems, fire rescue sensors deployed in rural or forest areas, sound

sensors deployed in forests to identify unauthorized human behaviours in forests, soil and plant condition monitoring sensors in farms, animal tracking sensors in farms and disaster identification sensor systems. So, the conclusion is sensor nodes have fewer data to transmit using less power. These sensor nodes are also equipped with less power consuming microcontrollers but those microcontrollers provide less processing as well. This happens because power consumption is relative to the processing capability. When transmitting data, it wants to transmit for long distances because sensors are deployed in far away. Hence a typical network like Wi-Fi is not suitable for such a sensor system. This is where it comes Low Power Wide Area Networks (LPWAN) to play. LPWANs also use for high battery consuming automotive devices like parcel delivering or security monitoring UAVs because they also have the same kind of requirement to transmit data for long distances consuming less power.

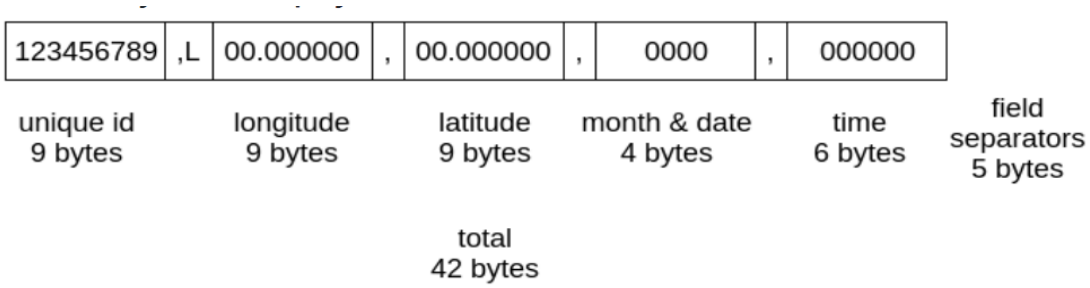


Figure 2: simple payload structure of a LoRa communication

A Low Power Wide Area Network uses a special kind of radio technology which supports transmitting for long distances consuming low battery power. Typically, this kind of networks does not support high data rates. So, such a network is ideal for sensor systems than typical communication networks like 2G, 3D, LTE, Wireline or Wi-Fi. **Figure 3** shows the growth of different kinds of networks for IOT over Time. It clearly shows that the growth rate of LPWAN is higher than other networks. Mekki, Kais, et al. [10] have compared about LoRa, SigFox, and NB-IoT which are today's leading and emergent radio technologies used in LPWANs.

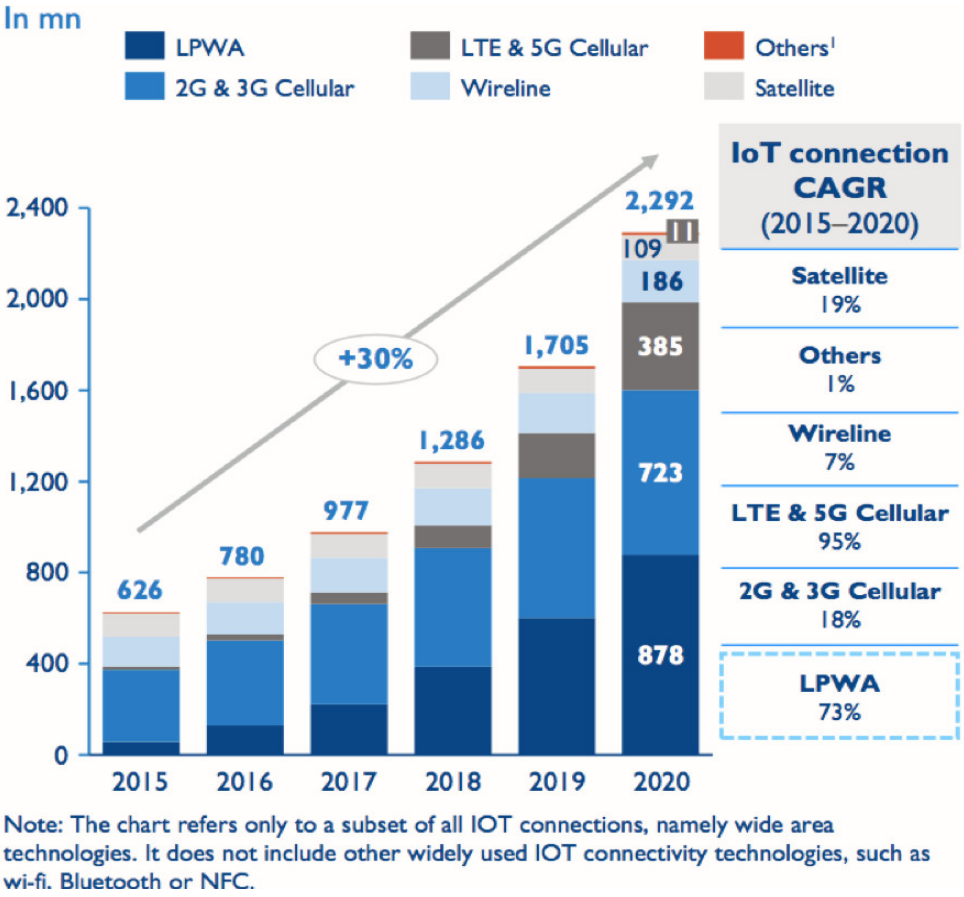


Figure 3: Population Wireless technologies and future expectation

LoRaWAN is such a Low Power Wide Area Networks which is widely spread all over the world. LoRaWAN [2][3][4] uses LoRa [1] which is introduced by Semtech as its radio technology and powered by LoRa Alliance. LoRaWAN architectures have sensor nodes, LoRa Gateways, Network servers, and Application servers. LoRa Radio is using between Sensor nodes and LoRa Gateways. Current statistics show that 4398 LoRa gateways and 47919 users have been registered in Lora Network [9]. **Figure 4** shows the spreading of LoRa gateways all over the world.

Figure 4: LoRaWAN usage over the world

1.2 Motivation

With the availability and sensitivity of sensor data, security has become an utmost important aspect for LPWANs. Sensor nodes are resource constrained devices and have to cope with its inherent limitations in electricity power, processing capability, and memory. However, implementation of security aspect on a sensor node may consume extra resources which are memory and power. This extra power consumption reduces the total alive time of a sensor node. Thus, it is important to manage these resources in an efficient manner while implementing security aspects for LPWANs. LoRaWAN is one of the widely spread secured Low Power Wide Area Network. However, it has its own limitations in security implementation. Security keys are important while providing the main security aspects authenticity, confidentiality, and Integrity. In LoRaWAN these security keys are bounded to the nodes at the time of firmware deployment and it cannot be modified dynamically. Therefore, in case of a security key revelation, the particular node should be physically accessed to update it with a new key. This problem can be resolved if there is a mechanism to distribute security keys dynamically which is known as Key Distribution in the information security domain.

1.3 Aims and Objective

LoRaWAN implementation uses hardcoded root key to generate session keys for secure communication of LoRaWAN protocol. Over-The-Air-Activation protocol is used to generate

those session keys using generated random numbers, EUIs and hardcoded root keys. This research aims to introduce an improved Key Management Protocol for LoRa Based Low Power Wide Area Networks considering the Limitations of End Node devices. When studying this objective, we have identified the following research questions.

a. What are the key distribution algorithms and protocols which can be applied to an LPWAN with its processing, memory, network and power limitations? [8] What is the most suitable key management protocol with those limitations?
b. Which implementation is the most suitable for the resource limitations of LoRaWAN Node.
c. How to validate the sensor nodes when exchanging the secret key?
d. How to improve existing security protocol with the new key derivation mechanism considering both power consumption and security level?
e. How and How often static keys should update in the system to be proactive to ensure the system security?

1.4 Research Scope

a. Software-based Key distribution mechanism for LoRaWAN network is going to introduce considering the capabilities of sensor nodes.
b. Key update and validation against key revelation will be discussed as a part of the key distribution mechanism.
c. Trade-Offs between the security level and resource limitations when exchange symmetric key will be discussed.
d. Mainly focus on key distribution and validation at LoRa communication part with the sensor nodes. The internet-based communication part is not focusing here.
e. Over the Air Activation mechanism and Class, A type messages protocol will be considered for this research.
f. Not going to implement a new encryption, authentication and integrity mechanisms for LoRaWAN.
g. The key exposition problems by physically accessing End Nodes are not going to solve.
h. The spread factor of LoRa radio technology varies from SF7& to SF12 and spread factor has an inversely proportional relationship to the Data Rate. All the improvements are going

to do for a selected spread factor (SF9). There are regional specific frequency ranges defined by LoRa community, in this research we are going to use 868MHz frequency range.

i. Different kinds of microcontrollers can be used in sensor nodes which are the End Nodes devices of the LoRaWAN. In this research, we consider only one device, LoRa32u4 II (Processor: 8MHz, Memory: 32KB, 3.3v) which is technically equivalent to the node introduces by LoRa Alliance.

1.5 Research Contribution

An improved and resource efficient key distribution and key management protocol for LoRaWAN instead of pre-bounded initial root key based key derivation of LoRaWAN protocol. This protocol will include key update, key rolling, session key generation, and End node validation.

1.6 Organization of Dissertation

Rest of this thesis is organized as follows, Chapter 2 provides a study of LoRaWAN using existing Literature. The latter half of Chapter 2 explains the literature review on Key Distribution algorithms and another approach to use them on microcontrollers which are used for Low Power Wide Area Networks. In Chapter 3 we describe the design approach and proposed the architecture of our research. Experimental Setup explaining data set used, algorithms used, tools used and evaluation metrics used are explained in Chapter 4. Next chapter, which is Chapter 5 describes our feature engineering process which is one of the pivotal components in our research. Chapter 6 provides Results and Analysis of the proposed approach and discuss the results summary. Chapter 7 concludes the thesis with a Conclusion and potential Future Work available of this research.

Chapter 2

Literature Review

This chapter extensively discusses the preliminary study of this research. In the first section, we explain LoRa and that why we specify the LoRaWAN among other Low power wide area networks which are briefly mentioned in the first chapter. Then we discuss specifications of LoRaWAN and Security vulnerabilities of LoRaWAN network. After that, we focus on the security key distribution problem and approaches taken by others to improve the key distribution. Finally, we approach the problem deeper by studying existing key distribution algorithms and How they can perform on microcontrollers which are work as the main board of Network Sensor Nodes.

2.1 LoRa

2.1.1 Overview of Lora

Lora is a radio technology invented by Semtech Corporation and introduced by Lora Allions. It uses chirp base modulation which is a kind of frequency modulation for transmitting bits. Data rate is very low in this radio technology because of the chirp base modulation. Lora technology is using all over the world and different regions allow different frequency bands for LoRa communication. There are three frequency bands using now 868Mhz ISM band for European countries, 433MHz for Asian region countries and 915MHz range for American region countries. Amount of data which can contain in a LoRa payload is limited. Message payload size is proportional to the time it takes to deliver the whole message (Time on Air). Lora has defined several spread factors (SF) to maintain a trade-off between message payload size to deliver and time on air (**Figure 5**). A frequency range has several LoRa channels and the Maximum number of nodes in a channel for the communication at a time is also limited. The Maximum number of devices in the channel depends on the payload sizes (**Figure 6**). Communication range of a device basically depends on the radio electronic component in the Lora Radio module and the power of antenna that uses with the Lora module. As an example, 20 dB power Lora radio module with a 5

dbi antenna supports for 3 km communication range in an open area. There are devices which can transmit for 10 km with an appropriate antenna. The data rate is the main limitation of Lora communication when applying Lora communication for the applications.

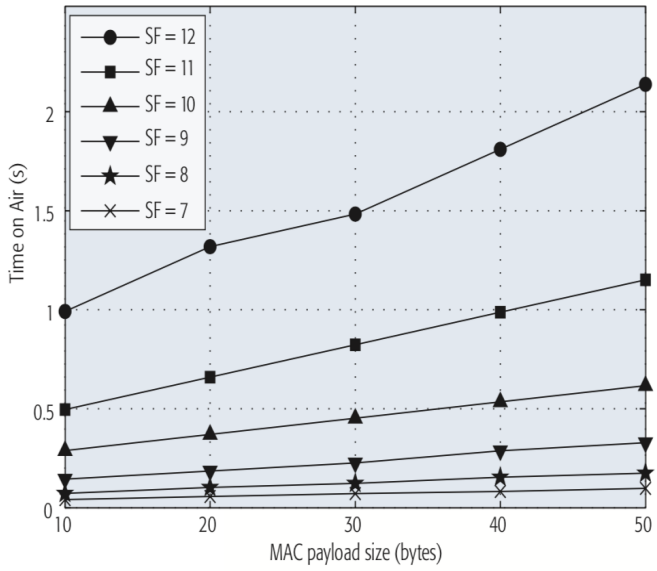


Figure 6: Time on Air over payload size for different spread factors

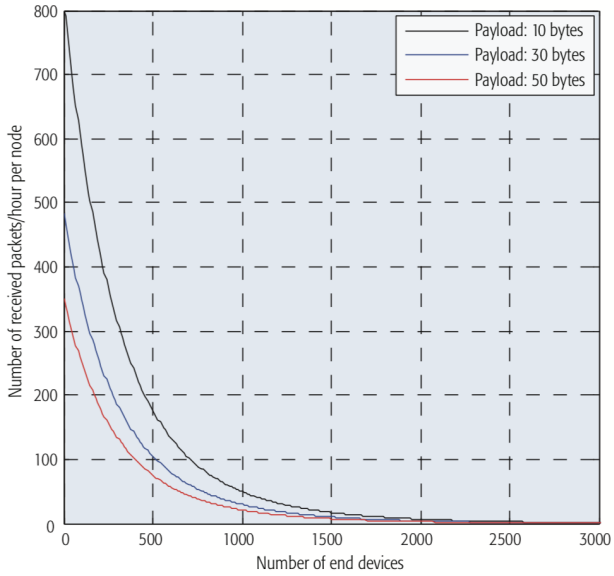


Figure 5: Data rate at a time over the number of nodes

2.1.2 Why LoRa for LPWAN

Low power wide area networks are sensor networks which consist less power consuming, long-range communicating sensor nodes. In kind of networks, transmitting distance and power consumption have an inversely proportional relation, hence when selecting a communication technology for an LPWAN we want to consider a feature analysis of this kind of networks. Mekki, Kais et al. [10] have compared about three communication technologies LoRa, SigPox, and NB-IoT which are today's leading and emergent radio technologies used for LPWANs. These three technologies are using three different radio modulation technologies. They have analysed Battery life, Quality of Service, Payload Length, Latency Performance, Scalability, Communication Range, Coverage, Deployment and Cost for LoRa SigPox and NB-IoT as shown in **Figure 7**. We want to agree for a trade-off because maximizing all the performance variables are not possible hence some of them they are inversely proportional to some other variables. For battery powered sensor networks power consumption is more considerable. According to this analysis both LoRa

and SigPox have similar power consumption for this research, we select LoRa because it uses as the radio technology of most widely spread LPWAN as we discussed in **Chapter 1**.

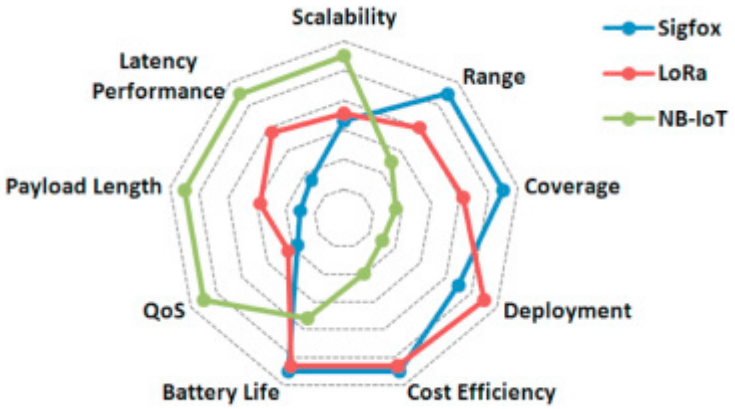


Figure 7: Respective advantages of Sigfox, LoRa, and NB-IoT [10].

2.1 LoRaWAN

2.1.1 Overview of LoRaWAN

LoRaWAN is the low power wide area network which uses LoRa as radio technology and deployed in more than one hundred countries. LoRaWAN is a proprietary product of Lora Alliance. LoRa Alliance™ et al. [2] specify LoRaWAN's newest specification of the implemented version. A Lora network has mainly four components of sensor nodes, Lora gateway, Network server, and application server. Sensor node and Lora gateways communicate using LoRa. From the gateway to Application server's communication happens through the TCP/IP. Network server manages the sensor nodes applications and gateways. **Figure 8** shows the Network architecture of the system with its main components.

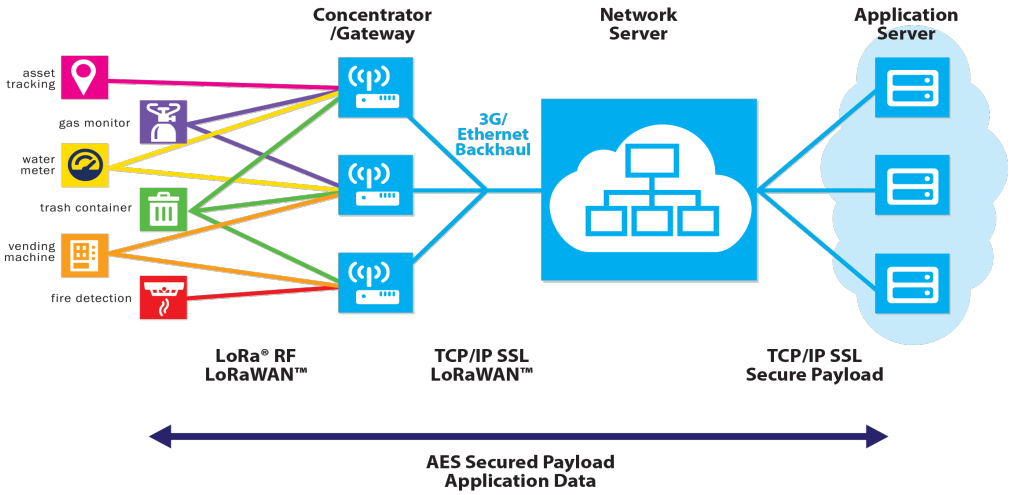


Figure 8 : Network Architecture of LoRaWAN [2]

LoraWAN consists of three message classes. Class A is the end to end bidirectional communication between Lora devices and gateways. Uplink is followed by two short downlink windows in this class. Class B is defined to receive time synchronized beacons from the gateways. End-devices of Class C have nearly continuously open receive windows, only closed when transmitting. Class C consumes more power than A and B classes. Only the Class A type messages are going to consider in this research for improving the security key distribution.

LoraWAN has three layers in its message protocol physical layer, network layer and transport layer and **Figure 9** explains the packet structures of different layers. In the physical layer, a packet contains a preamble, physical header, header checksum, payload, and checksum. This physical layer is handled by Lora modulation and it does nothing with LoraWAN. Network and transport layers handle by the LoraWAN. Physical layer payload may be MAC payload or joint request or join the response. This join request and response messages belong to device activation mechanism which discusses in the next subsection of this chapter. MIC is used for integrity checking and authentication at the network server. Frames handle by the application server for application-specific functionalities. Other than this message formats Newest specification of LoraWAN has discussed control messages which are exchanging between sensor nodes and Lora gateways for networking and controlling purposes.

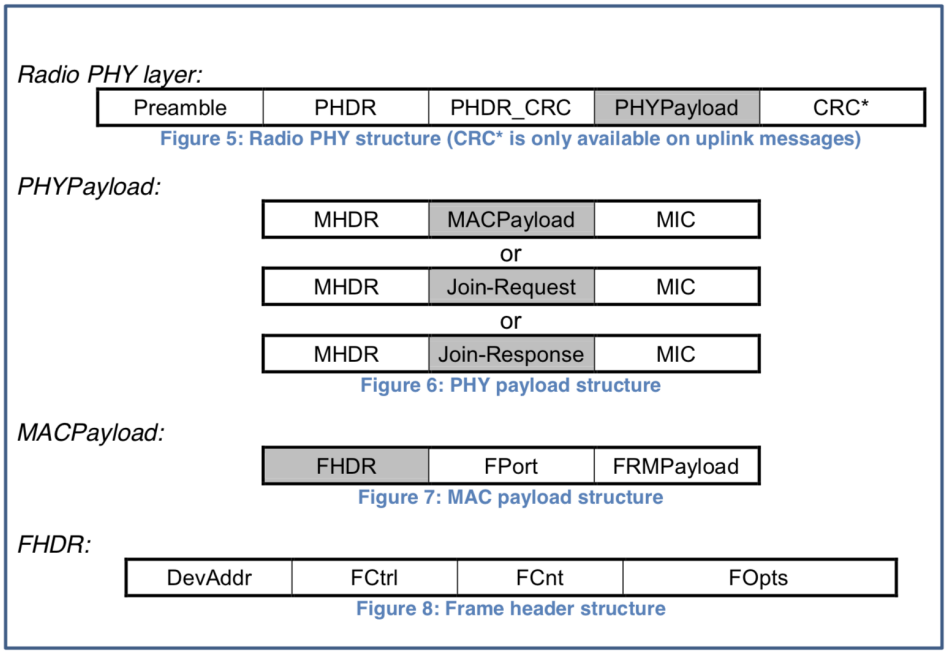


Figure 9 : Message formats in LoRaWAN [2]

2.1.2 Device Enrollment

Lora End Node devices want to join to the LoraWAN Network to communicate with the application server through the Lora Gateways. LoRa Alliance™ et al. [3] describes the backend network architectures and two End Device activation mechanisms Over-The-Air-Activation (OTTA) and Activate-By-Personalize(ABP). According to the LoRaWAN™ Specification v1.0.2 [2], OTTA mechanism the End node devices are deployed with AppEUI, DevEUI, AppKey. Here EUIs are 64bit identifiers to uniquely identify end node devices and Application. The AppKey is a pre-bounded root key to the End Node Device at the production level, This AppKey is private to the application and set up on the Network server by the application owner in his account. This AppKey is also used by the application server for End to End Security. The End Node Devices send a join request to Network Server for joining the network. This join request contains a randomly generated value (Dev Nonce), AppEUI and DeviceEUI. The request is digitally signed using AppKey as described in the function below for authentication and integrity.

$$Join\_request\_msg = AppEUI \mid DevEUI \mid DevNonce$$

$$Cmac = aes128\_cmac(AppKey, MHDR \mid Join\_request\_msg)$$

Network server reply to End device by join-accept response message which contains AppNonce, end-device address (DevAddr) along with configuration data for RF delays (RxDelay) and channels to use (CFList). This accepting response is encrypted and signed with AppKey using AES 128 Algorithms as below.

$$Join\_accept\_msg = AppNonce \mid NetID \mid DevAddr \mid DLSettings \mid RxDelay \mid CFList$$

$$cmac = aes128\_cmac(AppKey, MHDR \mid join\_accept\_msg)$$

$$aes128\_decrypt(AppKey, join - accept\_msg \mid cmac)$$

Typically, decrypting function consumes more power than Encryption function, hence network server uses decryption function to provide confidentiality and End-Nodes uses Encryption function to retrieve the plain message. Then End-Node devices can generate session keys to provide security aspects using AppNonce, DevNonce and AppKey.

$$NwkSKey = aes128\_enc(AppKey, 0x01 \mid AppNonce \mid NetID \mid DevNonce \mid pad)$$

$$AppSKey = aes128\_enc(AppKey, 0x02 \mid AppNonce \mid NetID \mid DevNonce \mid pad)$$

In Activation by Personalize (ABP) mechanism they don't have a joining mechanism, the session keys for confidential authentication and integrity purposes are pre-bounded to the End Node Devices at the production stage.

2.1.3 Confidentiality

Once a Node has joined a LoRa network, either through OTAA or ABP, all the messages should be encrypted using a security key to provide confidentiality. As specified in LoraWAN specification 1.0.2 [2] encryption is done using AppSkey from End Node device to the Application Server. NwkSkey is also used for encryption when it sends messages to the Network server. Only the network server contains the Nonces generates by End Node and itself. Hence application Server also able to generate the AppSkey through the Network server. Now, this AppSkey is held by three parties and it improves the probability of key compromisation we will discuss this later in this chapter. To provide the confidentiality it uses one out of these two session keys along with the

AES 128 Symmetric Key algorithm in Counter mode (CTR). AES CTR is a block cipher and it uses a counter value for the encryption instead of a chaining mechanism. An important feature of all messages in LoRa is that the counters for sent (FCntUp) and received (FCntDown) messages are maintained by the Node and Network Server and that these counters never repeat. For encryption and decryption, a keystream (S) is produced as follows:

$$i = 1..k \; where$$
$$k = ceil(len(FRMPayload) \; / \; 16)$$
$$Ai = (0x01 \mid (0x00 * 4) \mid Dir \mid DevAddr \mid FCntUp \; or \; FCntDown \mid 0x00 \mid i)$$
$$Si = aes128\_encrypt(K, Ai), for \; i = 1..k$$
$$S = S1|S2|..|Sk$$

The keystream includes the FCntUp or FCntDown values, which should mean that the keystream never repeats in the Node's lifetime. The FRMPayload is then XOR'd with the keystream to encrypt or decrypt the data. Other data such as the FPort and FCNTUp are sent unencrypted.

2.1.4 Integrity and Authentication

The MAC Payload section of messages are signed to prevent manipulation of the cipher-text, or of other values such as the DevAddr, FCntUp or FCntDown values. For join request message MIC is calculated using AppKey which is the hardcoded Key for OTAA mechanism as described in 2.12. Message integrity code for all the other messages are calculated using NwkSkey and Network server checks the integrity when message receives to detect unauthorized message manipulation. The 4-byte Message Integrity Code (MIC) is calculated as follows:

$$Msg = MHDR \mid FHDR \mid FPort \mid FRMPayload$$
$$B0 = (0x49 \mid 4 * 0x00 \mid Dir \mid DevAddr \mid FCntUp \; or \; FCntDown \mid 0x00 \mid len(msg))$$
$$mac = aes128\_cmac(NwkSKey, B0 \mid msg)$$

2.1.5 Replay Protection

Message replay attack which is also known as playback attack is network attack which transmits messages maliciously or fraudulently repeated or delayed. Lorawan uses message counters to

protect network over replay attack. For each End node device, there are two frame counters Frame Count UP(FCntUP) for uplink messages and Frame Count Down (FCntDown) for downlink messages. There is a limit value called Max Frame count Gap (MAX_FCNT_GAP) to keep sync in the uplink and downlink messages.

2.2 Security Vulnerabilities of LoRaWAN

LoraWAN has tried to provide the main security aspects confidentiality, integrity, authenticity, and availability up to a certain level we discussed them in the above section. Many people have studied about the security vulnerabilities of the LoRaWAN Network. Out of the Yang et al. [5] have analysed and categorized main possible attacks over a LoRaWAN Network into three categories confidentiality, availability and integrity as shown in **Figure 10**. They have examined replay attack for ABP-activated nodes, eavesdropping, Bit-Flipping attack, Acknowledgement spoofing, and LoRa class B attacks. They have also presented a proof-of-concept experiment for each of these attacks. Their suggestions to mitigate these attacks are briefly explained below.

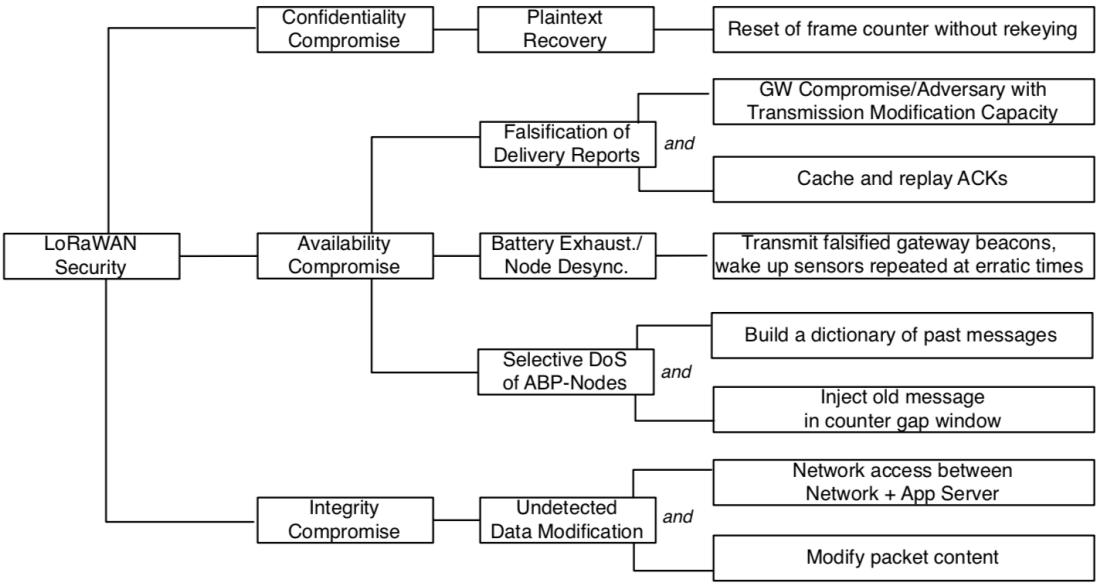


Figure 10: Vulnerability categorization of LoRaWAN [5].

- To prevent the **replay attack** ABP activated devices should use non-volatile memory as newly proposed LoRaWAN specification-1.1[4] which will be implemented in the future to avoid the frame counter resetting. For OTAA activated nodes, after overflowing the

counter, a replay attack is possible but doing the activation procedure again avoid this problem and moreover for ABP activated nodes devices can be reconfigured to change the session keys. However, reconfiguring nodes in a sensor network is not practical also for OTAA activated nodes key management is not practical when hardcoded root keys are revealed.

- **Eavesdropping** is possible while this AES block cipher uses a monotonically increasing counter. Using a random value will solve this up to a certain level but rekeying is needed when counter overflow is reached. Hence MCU's use for End node devices having a limited memory maintaining storage of pre-used nonces are also challengeable.
- Integrity checking is done at the Network Server using NwkSKey and decryption of payload at the Application Server using AppSKey but there is no integrity check at the Application server so **Bit-Flipping** attack is possible between Network Server and Application Server. This problem also is not solved by the new specification v1.1. They have proposed to implement another MIC for integrity checking at Application server to avoid this bit-flipping attack.
- Thus, the spread-spectrum technique use in LoRa with high spreading factor takes a long time for transmission. Hence selective jamming is possible with LoRa, then **ACK spoof** is can be easily done and newly propose specification v1.1 also exist this problem. And replaying previously recorded ACK's is also possible. To mitigate this problem, they suggest to include a cryptographic checksum to the ACK.

Other message classes of LoRaWAN also vulnerable to security attacks but we don't discuss them in here this research. E Aras et al. [6] also have explored the security vulnerabilities of LoRa. They discuss compromising the device and network keys, jamming techniques, replay attacks, and Wormhole attacks. They have concluded their exploration by highlighting the need for new key distribution mechanism and frame counter generation for AES CTR instead of pushing those to the developers end.

Woo-Jin Sung et al. [11] have analysed the replay attack in LoRaWAN network when joining an End Node Device to the Network via OTAA for both fixed devices and moving devices. Instead of identifying a device using the Dev-Nonce they have proposed a way to distinguish the End Node Devices from the attackers. Their suggestion is a lower level identification of devices using the RSSI as a countermeasure and more over a proprietary Hand-Shaking between End Node Devices and Application server. However, their proposition is needed to be verified in practice.

Hence the root keys are hardcoded in Devices and distributed with the Network server and Application server, joining procedure of the End Node Devices to the Network is the most vulnerable point in compromising the security keys for plain text recovery and also for other security vulnerabilities discussed above in this section. Several researchers have research on vulnerabilities of the OTAA mechanism and explore the improvements and mechanisms to secure the joining procedure. S Tomasin et al. [12] have examined the possibility of DoS attack due to the regeneration of an already used Dev-Nonce in the End Node devices. Possibility of detecting a replay attack at the Network server also have discussed by them. They have used SX1272 LoRa radio modem for their experiments and proposed the possible attack strategies.

Kevin Feichtinger et al. [7] also research on the vulnerabilities of Over The Air Activation of LoRaWAN and introduced a hybrid cryptosystem to encrypt the join request. This hybrid crypto-system uses already implemented a symmetric cryptosystem and hardcoded AppKey to encrypt the join request of OTAA mechanism. The regional differences of LoRaWAN ISM bands taken into consideration by them to ensure the applicability of their solution for all bands. They have suggested implementing and evaluate presented handshake by them to compare the performance with the existing mechanism. Furthermore, suggest a replacement of DevEUI of join-request with timing information as further researches.

LoRaWAN specification v1.1 [4] is proposed by LoRa to address the limitations of their latest deployed specification v1.0.2 [2]. In ov1.0.2 and previous specifications the Application Session key is handled by the Network server and also the Application key which is the hardcoded root key is also defined at Network Server. The main security keys have distributed among three parties, in v1.1 mainly they have introduced a set of new keys. Two root keys NwkKey and AppKey for Network layer and Application layer, three session keys are derived using these two root keys FNwkSIntKey, SNwkSIntKey for message integrity purpose and NwkSEncKey for confidentiality purposes of MAC command between End Node Devices and Network server. For previous uplink and downlink frame counters, they have introduced separate uplink counters for separate sources instead of one counter for all uplink messages in previous specifications. However, the Key Distribution and is still not handled by the LoRaWAN protocol because of two RootKeys.

This preliminary study of LoRaWAN shows the security vulnerabilities and the approaches of others to mitigate those vulnerabilities and attacks. As a summarization following options are available to improve the security in LoRaWAN protocol.

- Alternative cryptographic algorithm to AES in CTR to provide more confidentiality with considering the limitations of the End Node Devices.
- An alternative mechanism for security aspects of LoRaWAN instead of proposed new complex specification v 1.1 of LoRaWAN which is still vulnerable to the attacks.
- Explore the applicability of the suggested device activation mechanisms, improve them or new suggestions.
- Exploration of key distribution mechanism which supports to improve all the security aspects of LoRaWAN system and improve the maintainability of key deploying.

For this research, we are going to explore the possibility of using a key distribution algorithm to skip the joining mechanism with hardcoded root keys in OTAA. For these capabilities of End Node devices with their processing, memory and power limitations have to be considered in detail when selecting an already existing Key Distribution algorithm.

2.3 Key Management for LoRaWAN

2.3.1 LoraWAN 1.02 Joining Procedure

In section 2.1.2 of this chapter, we discussed the devices enrolment procedure of LoRaWAN network. ABP uses pre-shared session keys for secure communications. OTAA do key management by generating session keys based on the pre-shared root key which called as AppKey. Network server generates session keys for both itself and for the Application server. Hence no end to end communication between the sensor node and Application Server. **Figure 11** shows the message flow of OTAA joining mechanism of LoRaWAN specification 1.02. In section 2.3 we discussed the vulnerabilities of Join request and proposed solutions by the researchers. But the revelation of AppKey makesthat sensor to an unsecure state. Key revelation possibility is high because it duplicates at both sensor node and Network server. If network server attacked whole system become insecure. The main problem is it wants to access each node physically to update the Appkey. This not practical when maintaining the system.

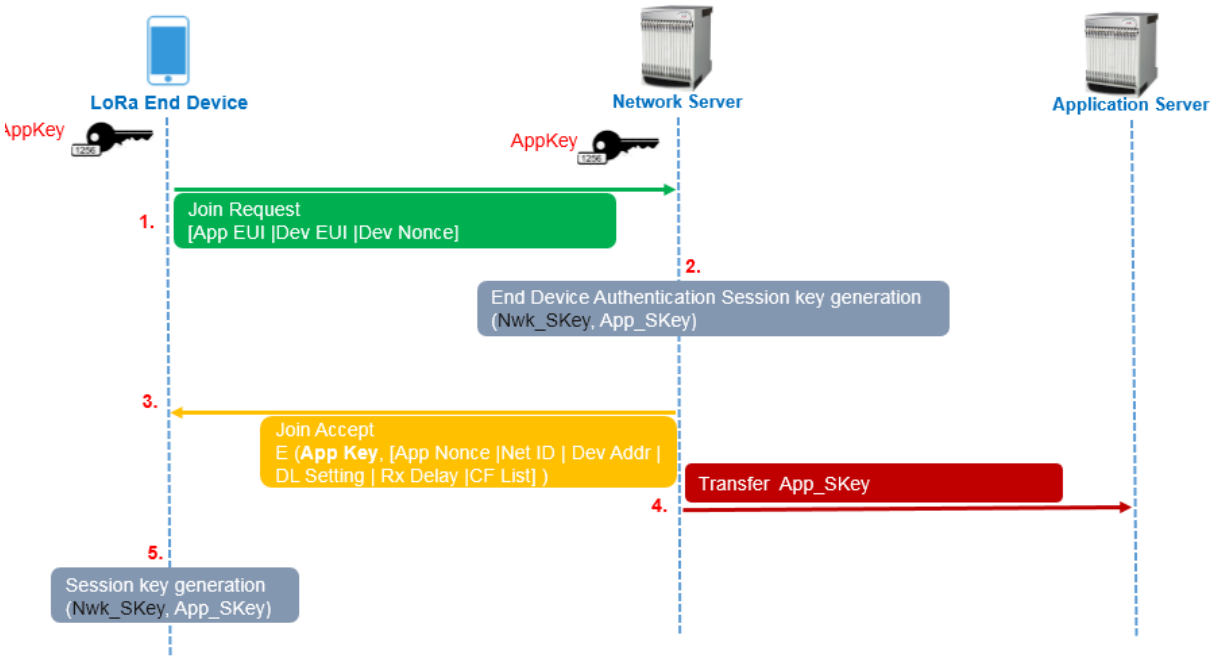


Figure 11: key exchange protocol in LoRaWAN 1.02

2.3.2 LoRaWAN 1.1 proposed Joining Procedure

Because of the vulnerabilities of existing key management of the joining procedure LoRaWAN system. LoRaAllience is proposing new version of the joining procedure by changing the entire system in their specification 1.1 [4]. They have proposed two root keys, NwkKey as root key for network server and AppKey as the root key of the application server. These two root keys are embedded to the Lora chip of the sensor node in the hardware level. These root keys are known by the Join Server which is newly proposed server for the LoRaNetwork. When Network server receives the join request it forwards the request to Join the server. Two session keys are generated at the Join server and deliver the relevant session key to the Network server and Application server. This key distribution is visualized in **Figure 12** as a high-level diagram. In this key distribution, all the root keys are store in Join server so this architecture may fall in a single point of failure to be insecure the whole system. This new architecture of LoRa Alliance is still not released as a production. LoRa chips with hardware level embedded keys and algorithms are still not available in the system. Migration of existing LoRaWAN networks to this architecture is also needed big effort and cost.

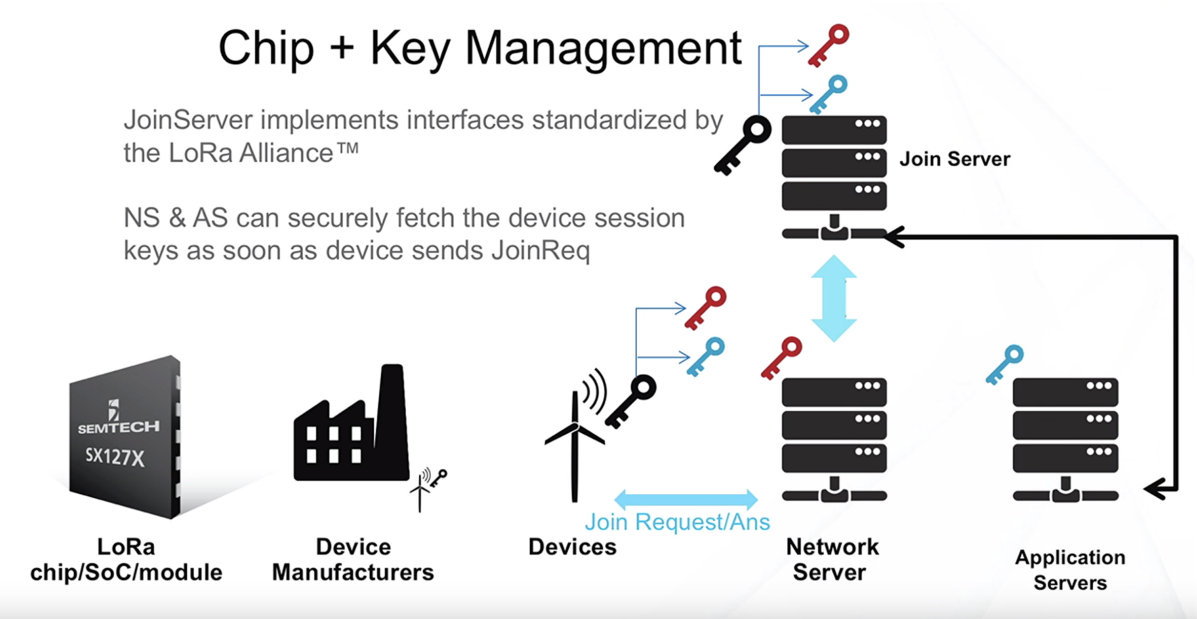


Figure 12: proposed key exchange protocol in LoRaWAN 1.1

2.3.3 Van Leent KDUM

Van Leent et al. [8] from the Cybersecurity academy of Netherlands also has studied the vulnerabilities of existing Sensor node joining procedure and key distribution. To mitigate those vulnerabilities, they raised the need for key distribution mechanism for the existing LoRaWAN architecture. They studied about LoRa, LoRaWAN existing key management and security status of the system. They have discussed different types of cryptographic algorithms including key distribution algorithms. They analyse the power consumption of key distribution and authentication algorithms by referring to the existing literature. They have selected the Elliptic curve Diffie Hellman (ECDH163r) algorithm for key distribution and Elliptic curve Digital Signature Algorithm (ECDSA163) as Digital Signature algorithm to verify the shared public keys. But the existing literature that he refers to analyse the key distribution algorithms are not relevant to the limitations that we met at LoRa end nodes. Most of the general cryptographic algorithms are not compatible with the 8-bit word size microcontrollers and their processing and memory constraints. They haven't discussed the strength of 163r curve to provide the equivalent for KD as existing strength of the system. Therefore, there is a need for an analysis of key distribution algorithms with the limitations of LoRaWAN End Devices to find a key distribution mechanism which suits for the limitation of Lora End Nodes. They haven’t discussed the key rolling

mechanism for the static security keys. The system should analyse the different scenarios where existing static key revelation is possible.

2.4 Summary

We started this chapter with the background study of the LoRaWAN system. We discussed LoRa technology and why LoRa suitable for wide area networks as radio technology. Then we discuss the LoRaWAN Network, its architecture how it works, different types of communications and so on. Hence this research focuses on security, then we move to discuss the security in LoRaWAN. We discuss sensor device enrollment to the network, how it provides main security aspects such as confidentiality, integrity, authentication and replay protection. Many researchers have conducted on the security issues of the LoRaWAN network, we have discussed the identified security issues by the research community. Several researchers have raised the importance of having a key distribution for the LoRaWAN and we discuss how key distribution handled in existing implementation and new proposals. Finally, we conclude the literature review by identifying the research gap of the proposed solutions for the key distribution of LoRaWAN Network.

Chapter 3

Methodology and Design

This chapter outlines the proposed solution and the experimental designs for LoRaWAN key exchange. We discuss the direction of our research based on the conclusion of the literature review. Then we decide what kind of approach we should use for this research to resolve the research questions. Variable identification for key distribution mechanism in the context of LoRaWAN with its limitations is going to consider. Then the high-level experimental design for the ECDH protocol evaluation will be discussed. And also, we introduce the high-level architecture of the key distribution mechanism which is proposed by this research.

3.1 Research Approach

LoRaWAN is a system for sensor networks which has a long range of communication capability with Lora chips on sensor devices. In chapter two we have discussed the security vulnerabilities identified by many researchers. Most of those vulnerabilities are related to an existing joining procedure which is called Over The Air Activation (OTAA) in LoRaWAN. Lora Alliance is introducing a new security approach for OTAA joining procedure [4]. Van Leent et al. [8] has also introduced a key management protocol for LoRaWAN which will mitigate the security vulnerabilities of existing joining procedure. They have analysed the performances of existing key exchange mechanisms and have selected *Elliptic Curve Diffie Hellman* (ECDH) [24] key distribution mechanism as the most appropriate key distribution mechanism for LoRaWAN. Their analysis has not discussed the capability of running ECDH on LoRaWAN sensor nodes which have limited processing, memory and power. We designed this research to evaluate the practical applicability of *Elliptic Curve Diffie Hellman* key exchange on LoraWAN nodes and to propose a key exchange mechanism to exchange keys between three parties which improves the system security.

There are three research approaches using in the research community, quantitative qualitative and mixed approach. Discovering and selecting a suitable ECC implementation is a performance evaluation of selected implementations of ECC algorithm by analyzing their resource requirement on Lora end nodes. Hence this analysis deals with performance measures on Lora nodes and this is playing with numbers, when considering the security, it is a qualitative fact but we can discuss the security strength of different ECC curves using existing literature which describe the strength using the key size used by the algorithm. Finally, we can categorize this part as a quantitative approach.

As the second part of this research, we propose a mechanism of key exchange between three parties of LoRaWAN system. Actually, this is based on the case study of the LoraWAN Sensor node joining mechanism and we try to propose a key distribution mechanism which improves the security level of the system. We discussed the security level of the mechanism which will be proposed by this research by building logic conditions using existing literature, then this part of the research can be considered as a qualitative approach of the research.

3.2 Methodology

Improving the system security is the objective of this research. We propose a key exchange mechanism for LoRaWAN Network considering its sensor node limitations to improve its security level. As Van Leent et al. [8] have presented the ECDH as the appropriate algorithm by analyzing the existing literature. Here we have considered the capability of running ECDH on LoRaWAN sensor nodes and applicability of ECDH to the LoRaWAN system. We broke down this work to several steps when achieving our goal.

1. Variable identification for key distribution on sensor nodes
2. Identifying relations between variables
3. Setup simple LoRaWAN system
4. Selecting existing implementation of ECDH
5. Experimenting to select the key sizes of ECDH
6. Designing key exchange mechanism between three parties
7. Implementing proof of concept of key exchange for LoRaWAN Node

After identifying variables and relations between them, we set up a simple LoRaWAN system with one End Node Device, one channel LoRa Gateway and one Application. We used this setup to identify component registration process and Node Activation process. In component registration, we identified Lora Gateway registration, End Node Device registration, and Application registration.

To integrate the ECDH base key exchange for LoRaWAN we want to implement the algorithm at two environments at special purpose embedded OS with fewer resources at Sensor Node side and general-purpose computers which not be a bottleneck for ECDH key exchange algorithms. There are various kinds of Elliptic Curves which have different resource consumptions, different security strengths. LoRaWAN End Node Device which is called as The Things Node [14] is powered with a microcontroller which has 8-bit word length. Most battery-powered sensor nodes use 8-bit word size microcontrollers. They also have relatively less processing power, less Random-Access Memory and also less flash memory. However, they consume less power consumption, hence they use for battery-powered sensor nodes. When selecting an ECDH implementation for LoRaWAN key exchange, we compared the performance analysis of different ECDH implementation using existing literature to select an ECDH implementation for LoRaWAN. Furthermore, we did experiments to measure the resource consumption of curve operations for the implementations which are still not in the existing literature. Flash memory consumption and memory for global variables are measured using the Arduino platform [44]. We measured Random Access Memory consumption by calculating the free space of memory using heap pointer and stack pointer. Instead of using electronic devices to measure power consumption, we measured the time consumption operations in the key exchange process. In cryptographic algorithms like Elliptic Curve operations they don't have any point of using sleep operations. For Elliptic Curve operations they also don't use GPIO[1] operations instead of ALU[2] operations. Then we can say the power consumption of operation is proportional to the time consumption of that operations under conditions mentioned above. We also considered security strengths like resistant to side channel attacks when selecting the Elliptic Curve implementation with the use of existing literature.

After selecting an Elliptic Curve implementation for 8-bit microcontrollers, we wanted to select a key size of the selected Elliptic Curve to provide sufficient security for the system. Using a longer key size consumes relatively higher resources. We set up experiments with the necessary variable

[1] GPIO:- General purpose input output which allow user to interact with computer using voltage inputs and outputs.

[2] ALU:- Arithmetic and Logic Unit.

and function definitions to exchange two secrets. Then we measured flash memory allocation, memory for global data, stack data and time consumption for each operation in the same way we described previously in this section.

Next step was to design the message flow to exchange two secret session keys between End Node Device, Application Server and between End Node Device, Network Server. We designed this key exchange to minimize the communications between End Node Device and other two servers because we need to minimize the power consumption of Key Exchange process to minimize the overhead on End Node Device and let it run for its maximum time. In the next chapter, we discuss the above steps in detail with experiments and results.

3.2 Design Concerns

We have proposed a key distribution mechanism for LoRaWAN and there are four parties involved in the communication of this system as we discussed in section one of chapter two under the overview of LoRaWAN. Sensor nodes, Lora Gateway, Network server, Application Server are the four parties and out of this four Lora Gateway switch the communication medium between Lora Radio and Internet. It blindly forwards all the received packets to the network server or to sensor nodes not being aware of the content. Sensor nodes send their data to the application server through the network server. Network server has the functionality of managing sensor nodes and Application server. Sensors want to communicate with both the Network server and Application server. But the application data should not reveal to the Network server.

As Van Leent et al. [8] discovered in their research, generally ECDH is better as an algorithm in performance for secure key exchange and management. But the ability to run ECDH on a sensor node and also exchange keys with two parties may cause an unexpected effect to the sensor node. Because the sensor node consists of a microcontroller which categorized as a specific purpose computing unit with limited resources. Sensor nodes have limitations on processing power, flash memory which store program data, SRAM and battery power. Hence using high resource consuming cryptographic algorithm like ECDH on sensor nodes is not straight forward as running them on general purpose computers. Limitations of sensor nodes are the main concern of this research when introducing the key exchange mechanism for LoRaWAN.

Basically, the ECDH algorithm describes the key exchange between two parties. But in LoRaWAN it participate three parties in communication. Therefore, we concerned the key exchange between three parties to ensure the end to end communication between sensor nodes and application server.

Authentication of end parties is also a need when exchanging the security keys. Hence authentication of sensor nodes was also a concern of this research.

3.3 Variable Identification

When performing the key distribution between a sensor node and another device, we wanted to consider the security level and the limitations of the sensor node. For dealing with this, we identified system variable which involves key distribution on LoRaWAN sensor nodes.

- ECDH implementation for 8-bit microcontrollers
- Share secret
- Symmetric key
- Security level
- Power consumption
- Memory consumption

These variables can categorize to two parts, independent variables and dependent variables. **Figure 13** describes the variable categorization and their high-level relations.
Purpose of using a key distribution mechanism is to share a secret key between two parties. Then using this shared secret, both parties can generate a symmetric key for security purposes like data encryption. Symmetric key can be generated or derived from shared secret using a different kind of Algorithms. When providing security, ethic is hiding the secret keys instead of hiding algorithms. Hence the system security totally depends on the shared secret, we considered both share secret and symmetric key as independent variables. Other three variables which are security level, power consumption, and memory consumption depends on the shared secret size which is shared using the ECDH protocol. To have different shared secrets we wanted to use different curves ECDH algorithm. Different curves provide different security levels and resource consumption is also different.

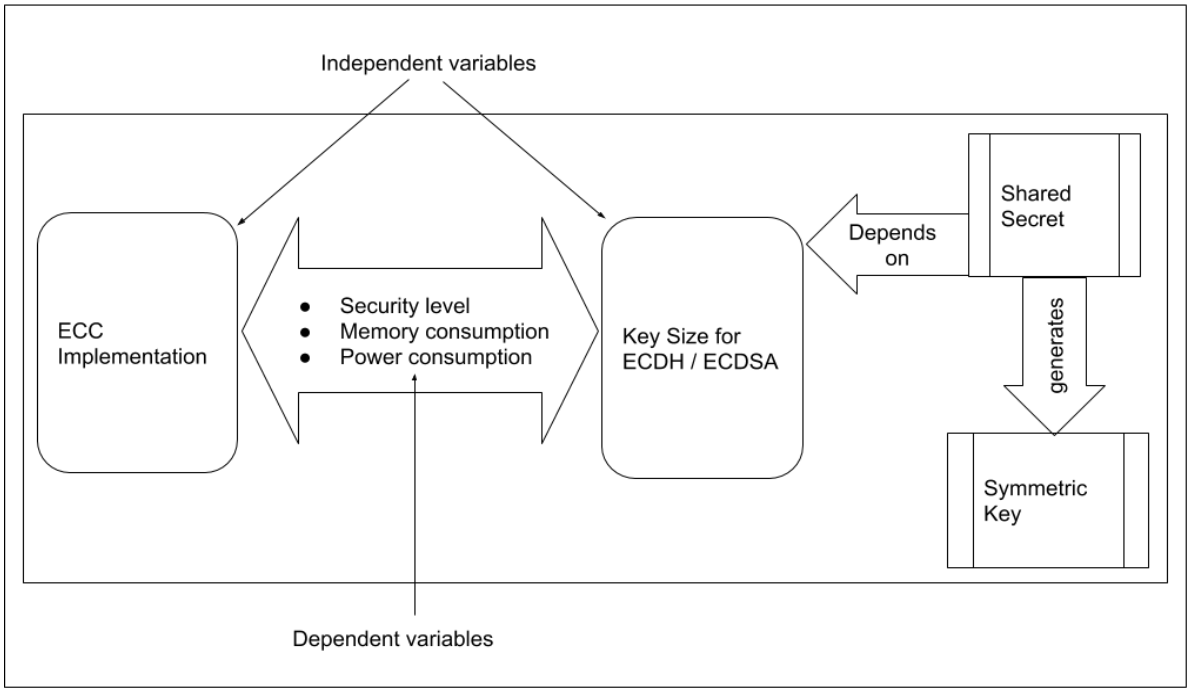


Figure 13: Identification of dependent and independent variables of key distribution and their relationships.

After selecting the shared secret size, we wanted to design key exchange between three parties. Sensor node should communicate with both Network server and Application server bidirectionally. When designing the key exchange between three parties we considered three dependent variables memory consumption, power consumption and security level.

3.4 High-Level System Architecture for Key Distribution

With the result of the experiments, we designed a Key Distribution mechanism which uses ECDH as the base cryptographic algorithm. Here with this Architecture, we propose the algorithms, different types of keys, main variables for each party take part in LoRa Key Distribution. And we present a message flow for efficiently distribute the secrets between sensor node, network server, and the application server. Here we present the high-level architecture for LoRa WAN key distribution as visualize in **Figure 14** and at the end of this research, we present the architecture in detail. In high-level architecture, a sensor node takes part in two key distributions one for the

Application server and another one for the Network server. Sensor node should contain its private key, the public keys of all including itself and a shared secret for each other party. Lora gateway is also a party take part in Lora communication but it doesn't take part in key distribution protocol. Its functionality is to forward packets blindly between two networks Lora radio and internet.

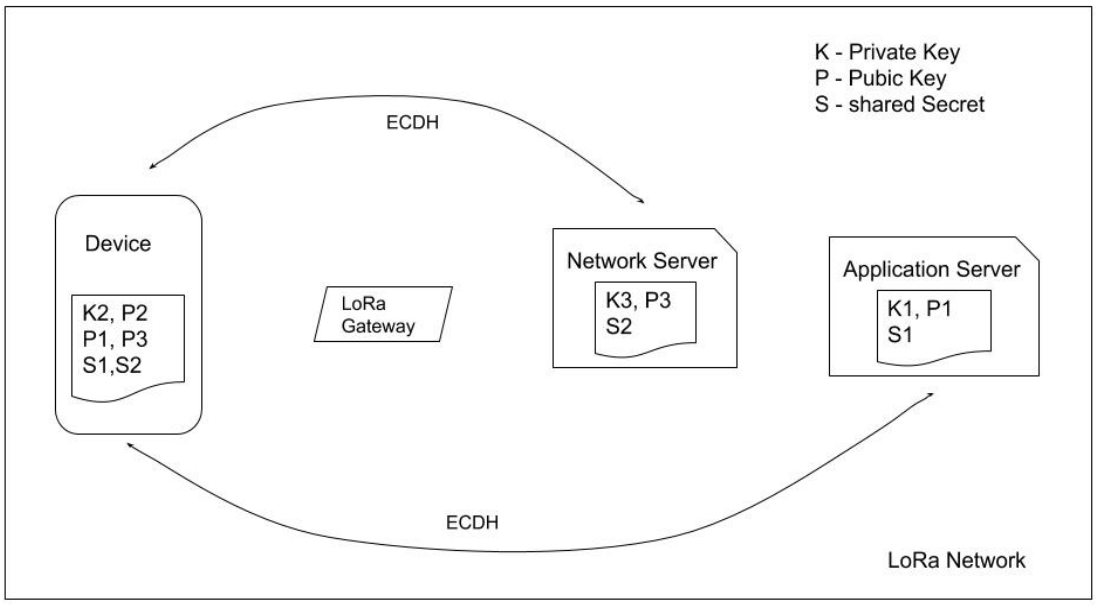


Figure 14: Suggested Key distribution for LoRaWAN

3.5 Summary

In this chapter, we discussed our approach for research on using ECDH as the cryptographic algorithm for Key Distribution in LoRaWAN. We stated with the facts found in the literature review and built our approach. Then we discussed how we identified the system variables, their categorization, and relations. We briefly explained the methodology and the steps of reaching the goal. Finally, we proposed the high-level key distribution architecture for LoRaWAN Network using the ECDH as the cryptographic algorithm to ensure all the security aspects of the system.

Chapter 4

Experiments and Proposed Solutions

4.1 Relationships between variables

We identified the variables of the LoRaWAN system when providing the data security of LoRaWAN communication. To secure the sensor data, AES encryption is used in the existing architecture and a common key which is called as a symmetric key should be shared between the parties who are going to encrypt or decrypt the data. Using Elliptic Curve Diffie Hellman key exchange we can share a common secret between two parties. Then we can generate the session key at both parties using the shared secret. To generate the session key, we can use an algorithm such as hashing or simply we can take the first N bits of the shared secret if the shared secret is larger than the expected symmetric key. However symmetric key generation using the shared secret cannot improve the system security. We cannot use any other secret in this key generation because then sharing this other secret would be another problem. With this argument, we can decide that the security level of the symmetric key is relatively proportional to the security level of the secret which shared by the key distribution mechanism. We summarise these arguments which can be considered as the facts in data security in the following two equations.

- $Data\ security\ of\ a\ LoRaWAN\ message\ \ \alpha\ \ Key\ size\ of\ the\ symmetric\ key$
- $Security\ level\ of\ the\ symmetric\ key\ \ \alpha\ \ Security\ level\ of\ the\ shared\ secret$

Our main focus is to test the ability of ECDH with the sensor node limitations, all variables that we identified should be tested with the sensor device. We want to maximize the security level LoRaWAN system. Limitations at the sensor node is the bottleneck of the system in power, memory and processing. Hence, the sensor node may be a barrier when maximizing the security level of key distribution. When considering the Elliptic Curve Cryptography there are several standard Elliptic curves. Performance and security features are different from curve to curve. When considering a specific curve, we can use it to share different sizes of secrets. The public key and private key pair directly related to the shared secret. Running an Elliptic curve cryptography algorithm consumes relatively a more power of sensor node and a considerable amount of memory.

Both power and memory consumption relatively proportional to the key sizes using in the algorithm. And also, both power and memory consumptions depend on the selected curve implementation. Following two equations represent the proportional relationship of key sizes and curve implementation to power and memory consumption.

- $Power\ consumption\ for\ key\ distribution\ \alpha\ \ keysize\ \&\ implementation$
- $Memory\ consumption\ for\ key\ distribution\ \alpha\ \ keysize\ \&\ implementation$

In this research, we select suitable implementation of ECC for 8-bit microcontrollers using existing both existing literature and our experiments. Then the selection of the key size for the selected implementation will discuss with our experiments. The intention of selecting both implementation and key size is having a trade-off between security and resource consumption.

4.2 Setup LoRaWAN system

LoRaWAN existing system has four main components as we mentioned in previous chapters. They are Sensor Nodes, LoRa Gateway, Network Server, Application Server. The network server is provided by them and we need to set up other three components and register them in their server which is the network server. First, we need to Register our Application in the Network Server. We should select a message handling service URL and should add a unique identifier for the Application which called Application EUI. Handler service is the service which does the message exchange service for the network server. LoRa has setup different handler services for different regions. This EUI should be Hardcoded in each End node as well as in the Application Server. It ensures the communication between the Application and the relevant End Nodes. Network server forward received packets by filtering them using their Application EUI. **Figure 15** shows the details of the already registered Application in the Network Server. See Appendix 1 for the Joining details of Devices at the Network Server.

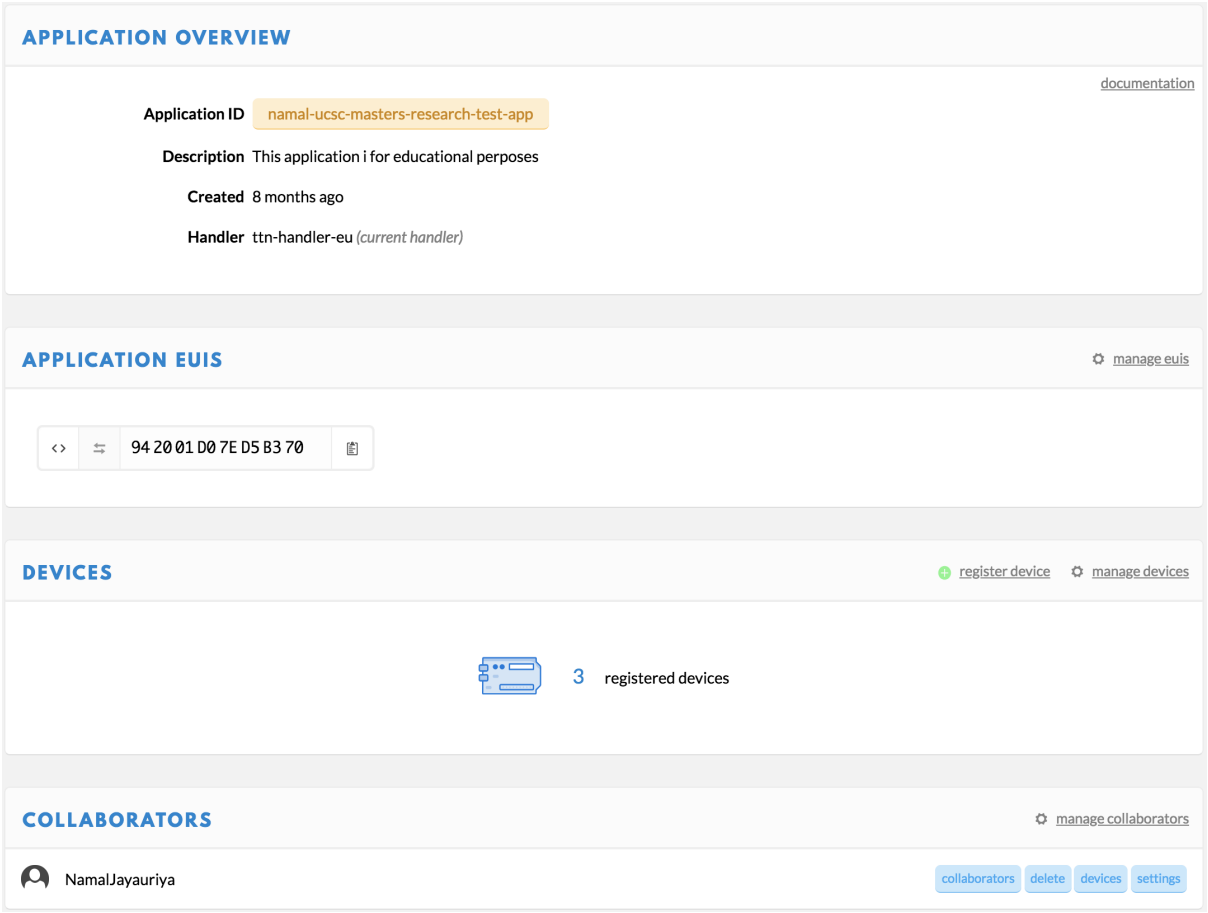


Figure 15: Overview of a registered Application

After registered the Application We can add the devices for the Application. We want to set Application EUI, Device EUI and App Key. Application EUI is common for an Application and Application EUI and App Key should be unique for each End Node Device. When registering a Node, the joining mechanism (OTAA or ABP) also should be selected. For ABP devices Network session key and Application session key should add instead of App Key. For OTAA devices, Network session key and Application session key are generated when a device is joined to the network. For both joining methods device MAC addresses are received when they connected to the Network server. Settings window of a Device which in **Figure 16** shows the variables we set at the registration of the Device.

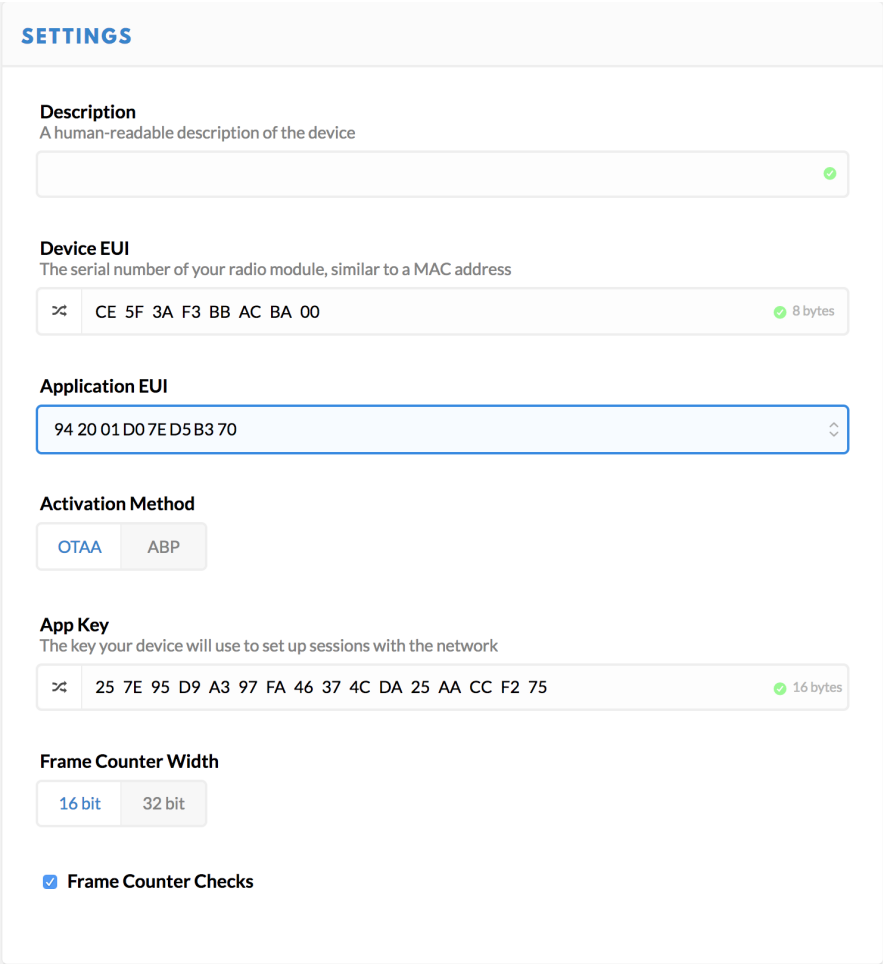


Figure 16: Settings window of a registered Device

In this research, we investigate the capability of running ECDH base key exchange on Sensor Node. Application Server is a computer which may be a server version or a general-purpose computer which has enough resources for cryptographic algorithms of Key Exchange mechanism. Hence, we do not need to focus on the Application Server side and we are not going to Implement and setup an Application server.

A LoRa gateway needs to switch the messages between two networks LoRa network and the internet. The Things Gateway [25] introduce by LoRa Alliance support for multi-channel and multi-frequency bands and cost also high for these gateways. We do not use End Node Devices of different frequency bands. Hence, we setup a single channel gateway for our experiments. We use ESP8266 [26] wifi embedded microcontroller chip as the processor and RFM95 [27] LoRa radio chip which supports 868 MHz frequency range for our LoRaWAN gateway. We use ESP Single Channel Gateway library for Arduino as the firmware of our gateway. We set the Things Network Network server IP to forward the received LoRa packets to the Network server. We set the SSID

and password of a wifi network to connect the gateway to the internet. This single channel gateway also has a tiny web server to set the configurations and we use that to set the configurations like LoRaWAN spread factor. All the settings we use at our gateway shows at the **Figure 17.**

Setting	Value	Set
CAD	OFF	ON OFF
HOP	OFF	ON OFF
SF Setting	9	- +
Channel	0	- +
Debug level	1	- +
Debug pattern	SCN CAD	RX TX
	PRE MAI	GUI RDIO
Usb Debug	1	
Framecounter Internal Sensor	0	RESET
Gateway Node	OFF ON	OFF
WWW Refresh	OFF	ON OFF
Update Firmware		UPDATE
Format SPIFFS		FORMAT
Statistics	0	RESET
Boots and Resets	1	RESET

Figure 17: Single channel gateway configuration interface

For the sensor devices, LoRa Alliance introduces a sensor node which is called as “The Things Node” [14] and a breakout board called “The Things Uno” [13] for designing the sensor nodes. LoRa Alliance design application-specific custom sensor nodes for user requested designs with the required sensors for user applications. Both devices consist of the same microcontroller Atmega32u4 [15] and same radio module RN2483 [16]. For our experiments, we use LoRa32u4II [18] breakout board which consist of the same microcontroller but consist of a different LoRa radio chip which is SX1276 [17]. LoRa32u4II have 28672 Bytes flash memory for code and data space, 2.500KB of Static RAM, 1KB of EEPROM, 8MHz clock speed at 3.3V and 8-bit word size. SX1276 radio chip that uses in LoRa32u4II is low in cost in the market and support for low communication ranges with compared to RN2483 chip. Both of these radio chips use LoRa modulation as the data modulation technique and this research consider the capability of running the Key Exchange mechanism using ECDH. Hence using a different radio chip of a different manufacturer doesn’t affect for our experiments. **Table 1** summarizes the differences of LoRaWAN sensor node and the Node we use in this research.

device	Microcontroller	LoRa Radio chip
The Things Node	ATMega32u4	RN2483
The Things Uno	ATMega32u4	RN2483
LoRa32u4 II	ATMega32u4	SX1276

Table 1: Compare End Node which we use with the Node provide by LoRaWAN

Then we want to configure the End node devices with the parameters which we set at the Network server. We use the Arduino-MCCI library [28] version 2.3.2 for the firmware of the sensor node device which is an extended version of the IBM Arduino-LMIC library [19]. This library supports LoRaWAN 1.02 specifications which are the current product ready version of LoRawan. We Setup LoRa Node with Spread Factor 9, LoRa channel '1' as the LoRa radio configurations. LoRaWAN facilitates different communication features, but we do our experiments with minimum features which must be there to continue communication between LoRaWAN components. MCCI library allows users to configure the project by modifying 'project_config/lmic_project_config.h' at the root directory of the library and we refer the configuration instruction in library `README.md`. The microcontroller used in Sensor Node executes the processor instructions sequentially. When no interrupts occur, extra features of LoRaWAN do not cause for Joining procedures power consumption and stack memory allocation. But these features cause for overall power consumption of the Sensor Node. However, these extra features of LoRaWAN cause for global data space and flash memory space. We disable the extra features of MCCI implementation and measure the flash memory and global data usage for OTAA of the Things Node.

We do the experiment to measure resource consumption of Session key exchanging process of existing implementation which is LoRaWAN specification 1.02. We use Arduino IDE to measure the flash memory allocation and memory allocation for global data as we discussed in the Methodology section in chapter 3. First, we disable the joining by defining "DISABLE_JOIN" in lmic_project_config.h header file under the "project_config" directory of MCCI library and measure memory allocations. Then we do the same experiment with enabling the joining. We use "Memory Free" library [37] to measure the main memory allocation at run time. We measure free space of RAM at initiate joining, send join request, join success and send data and we calculate RAM allocation at each time. We discuss the obtained results in the evaluation chapter by suggesting the key exchange mechanism.

4.3 Selecting ECC implementation for LoRaWAN

Van Leent Analysed performance of cryptographic algorithms such as RSA [36], Diffie Hellman [38], Elliptic Curve Diffie Hellman (ECDH) [24] using existing literature as we mentioned in the literature review. They suggest having a new key distribution mechanism instead of OTAA in LoRaWAN using ECDH algorithm. They also suggest using ECDSA [39] algorithm to avoid the vulnerabilities of exposing to the man in the middle attacks. Now it has different types of elliptic curves and different implementations. Some of them are very high in resource consumption.

When integrating ECDH key exchange to LoRaWAN, first we need to check the resource availability of the system for ECDH. We consider RAM space consumption, time consumption and flash memory consumption to check the resource availability. It is sufficient to check the resource availability at the End Node devices which is the resource bottleneck of the system. When selecting the ECC implementation for LoRaWAN we first check the flash memory availability to integrate different implementations of ECC. LoRaWAN stack implementation, application implementations, key exchange implementation and hardcoded variables all should be stored at the flash memory. we check the available code space after allocating space to LoRaWAN stack and there should be enough space for application code after implementing the key exchange.

After selecting the implementations, we check the availability of RAM to run the system with LoRa stack, key exchange and application. This type of microcontrollers executes the processes sequentially and we disable the LoRaWAN interrupts for our experiment setup. Hence key exchange \does not run parallelly with other processes and stack allocates only for one process at a time. However global data for all programs allocate some SRAM memory space. The remain RAM space is available for the running process. We check whether the remaining RAM is enough or not for the key exchange processes and select matching implementations out of previously selected implementations.

Finally, we consider the time consumption for key exchange for different key sizes of the selected implementations to select the better implementation of ECC for LoRaWAN. The execution time of ECDH cause for the battery life of the Sensor Node because cryptographic computations consume more processing power. However, Key Exchange is used to share a symmetric key and this operation does not happen frequently because of that taking a few seconds to exchange keys is also bearable.

Zhe Liu et al. from a Microsoft research group propose FourQ [20] as a more efficient and secure implementation of Elliptic Curve. They have used IAR Embedded Workbench [40] which is a

proprietary development platform for their testings and evaluations with 8-bit, 16-bit and 32-bit microcontroller architectures. We consider the experiment results which target 8-bit microcontrollers because in this research we target 8-bit microcontrollers. As the 8-bit microcontroller, they target ATxmega256A3 [45] which consist of more memory and processing resources, 32 MHz clock speed, 256 KB flash, 4 KB EEPROM, 16 KB SRAM. However, this microcontroller is power consumable than ATmega32u4 which we use in LoRaWAN nodes. Zhe Liu et al. have analysed and evaluated performance and memory consumptions of their implementation "FourQ" with Curve25519 [23] and μKummer [41] ECC implementations. We use their analysis and our experiments to select the ECC implementation for LoRaWAN.

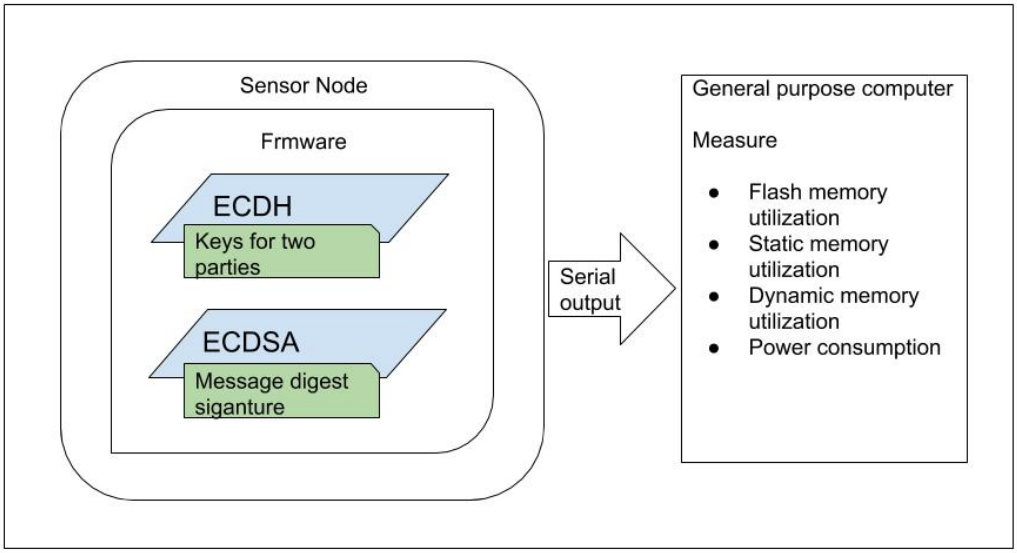


Figure 18: ECDH and ECDSA algorithms on the Sensor node

We set up an experiment to test the ability to run ECDH on LoRaWAN Node. Design of the experiment shows in **Figure 18**. We test micro-ecc [22] which is an implementation of standard SECP curves [29], nano-ecc [42] which is a fork of initial micro-ecc and also curve25519. Our experimental setup consists of a LoRa32u4 II module connected to a pc via a serial connection. Here we set up with the minimal requirement to test the running capability ECDH key exchange, ECDS, and ECDS verification on the same node. Hence, source code burned to the microcontroller with variables for two key pairs, signature, verification string and a message string.

We consider the memory consumption to select the ECC implementation for LoRaWAN. We use the results of Zhe Liu et al.'s FourQ evaluation and results of our experimental setup discussed

above. All the results show that remain flash memory space after setting up LoRaWAN is not enough for any implementation. Hence, we need a new implementation of the LoRaWAN stack which consumes less memory. However, we select micro-ECC library implemented by Kmacky which implements the SECP curves for different key sizes as the suitable ECC implementation for LoRaWAN Nodes to try with LoRa stack because it consumes less code space compared to other implementations. We discuss the results which we use to evaluate and selection of micro-ecc implementation at the evaluation chapter.

4.4 Selecting ECDH key sizes

An Elliptic Curve implementation can support one or more key sizes. As an example, curve 25519 only supports 256-bit private key size and Kmacky's micro implementation supports 4 key sizes. Configuring an implementation of an ECC implementation with different key sizes are also considered as different curves. Different key sizes provide different security levels and the security level is proportional to the key size. OpenSSL also uses Elliptic Curve Diffie Hellman in sharing the session key for transport layer security (SSL/TLS). OpenSSL wiki [21] discuss the security levels of asymmetric key cryptographic and Elliptic curve cryptography comparing with symmetric key security levels. The standard of measuring Security level of asymmetric key and ECC cryptography is comparison with the symmetric key security levels. As an example, security level of 3072 asymmetric key sizes and security level of 256 Elliptic Curve key size are equal to the security level of 128-bit symmetric key security level. A symmetric key security level is measured with the time takes to brute force the key. The standard comparison between Asymmetric, ECC and Symmetric keys is shown in **table 2**.

Symmetric Key Leng	Standard asymmetric Key Length	Elliptic Curve Key Length
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	512

Table 2: key strength comparison of symmetric, asymmetric and Elliptic Curve

LoRaWAN provides 128-bit confidential level for its sensor data. Hence, key distribution should provide equal or higher security level in key exchange mechanism and otherwise, the symmetric key will be breached easily through key exchange mechanism than brute forcing it. We have selected micro-ecc as a suitable implementation for the LoRaWAN End Nodes. Micro-ecc consists of two types of elliptic curves for 256-bit key sizes, Koblitz curve and random curve. In SEC2 standard these two curves names as secp256k1, secp256r1 respectively. Micro-ecc also supports other key sizes 160, 192, 224 bits as random curves. At the Evaluation chapter, we discuss the selection of key size for LoRaWAN key exchange.

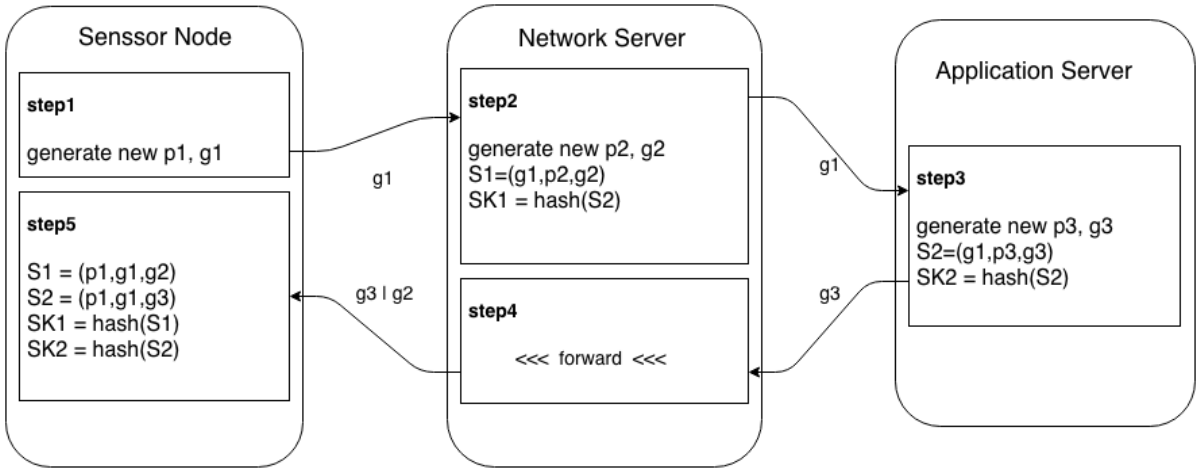
4.5 Key Distribution Mechanism

4.5.1 Key Exchange

In LoRaWAN architecture we register Our Applications and End Node devices at the Network server. Then the Network server maintains the communication between End Nodes and Application server. Receive packets from End Nodes through Gateways, Handle End Node joining and re-joining. In existing OTAA joining mechanism End node sends a generated random number

to the Network server. Then Network server generate another random number and send it back to the End node. Then both parties generate two session keys and Network server also sends the generated session key of the Application Server to the Application Server. At the literature review, we discuss the vulnerabilities of this existing OTAA and proposed solutions to it. Using a key exchange mechanism is one proposed solution to increase the security of Joining mechanism.

In our research, we have selected a suitable implementation of ECC and a key size for the selected implementation. Then we need to introduce a key exchange mechanism using Elliptic Curve Cryptography. We want to introduce the message flow between LoRa End Node and both Application Server and Network Server. Our aim is to reduce the resource consumption at the End Node device. Sensor node has very limited memory space and also limited battery power. Hence, we should reduce code lines, static variables, run-time variables to make this memory efficient. Also, executions of cryptographic functions and the number of communications for key exchange should be reduced. Initially, when a node joins the network, the node should be registered at the Network server under the relevant Application. They use 64-bit Extended Unique Identifiers (EUI) to identify the End Node Devices and Applications. After End Node Devices Deployed, it wants to initiate the communication from the End Node Device side. To have secure communication between three parties, we want to exchange two secrets from the sensor node. As mentioned in the scope we assume that we have already secured communication channel between the Network server and Application server. According to LoRaWAN architecture sensor node communicate between Application Server through the Network server. Hence, we try to exchange two secrets using a single communication cycle. First, Sensor Node generates an ECC key pair and shares the public key to both Network server and Application server. Both Application server and Network server generates keys pairs and send their public keys as a single message to Sensor Node through the Network Server. Then Sensor Node can generate two secrets for two servers. LoRaWAN stack needs a 128-bit session key. We can use a hash algorithm or simply first 128 bit of shares secret as the session key. **Figure 19** shows the message flow of the proposed key exchange mechanism for the LoRaWAN network.



P1 - ECC private key of End Node
g1 - ECC public key of End Node

P2 - ECC private key of Network server
g2 - ECC public key of Network server

P3 - ECC private key of Application Server

S1 - shared secret between Node and Network server
S2 - shared secret between Node and Application server

SK1 - session key between Node and Network server
SK 2 - session key between Node and Application server

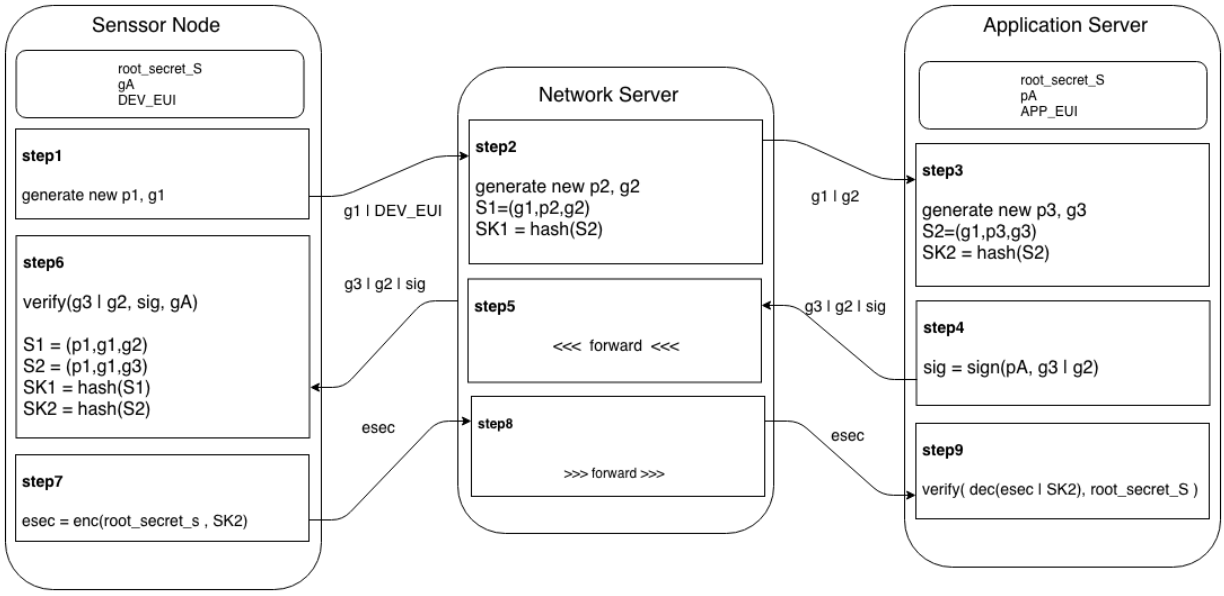
Figure 19: Initial key exchange between Sensor node and servers.

4.5.2 Node Validation

When exchanging secrets to generate session keys we should authenticate all the parties take part in the key exchange. Otherwise bogus parties can act as real parties. In general, Diffie Hellman Key Exchange is vulnerable to man in the middle attack. Hence, ECDH also vulnerable to man in the middle attack. Authentication mechanisms can use to prevent the man in the middle attack also. With Elliptic Cryptography, Elliptic Curve Digital Signature Algorithm can be used to authenticate a party. In LoRaWAN System we want to Authenticate all three parties, End Node Devices, Application Server and also Network Server. Typically, public keys of End parties are authenticated with public key certificates which are issued by Trusted Certification Authorities. But for LoRaWAN System it is not practical to use public key certificates hence the nodes are self-survived.

When a new Node Deployed, it initiates communication by doing the key exchange. At this time sensor node want to authenticate its identity to other parties. Hence, Application server is already

authenticated to the system, it is enough to authenticate the identity of End node to one of two servers. Network Server is a third-party common Server for many applications and there should be an end to end communication between Sensor Nodes and Applications Server. Hence, we decide to authenticate all the Sensor Nodes to its application server. If authentication of a node fails, then Application Server can request to the Network server to ignore that End Node Device. We introduce a unique root secret for each End Node Device then this should be known by the relevant application. The revelation of a secret of a node does not affect other nodes or servers but revelation root secrets at Application Server effect for all End Nodes. As a solution for this, at the Application server side, a Hardware Secure Module (HSM) [33] can be used to secure all the root secrets of end nodes. To authenticate the identity of the two Servers, we introduce a root elliptic curve key pair for the Application Server. The public key of the Application Server key pair wants to share with all the Sensor Nodes relevant to that Application. The revelation of the public key at a sensor node is not a problem because it is the public key but revelation of the private key of Application needs to update all sensor node and Application to re-setup the security. **Figure 20** describes the steps of key exchange with authentication. First Sensor Node generates a key pair and sends public key and Device EUI to the servers. Here we send Device EUI to identify the Node at the Network server then it can redirect the message to relevant Application Server. Then key pairs generated at two servers and also secrets generate using Sensor Nodes public key. At the Application server concatenation of public keys of both servers are signed and send to the Sensor Node. Then Sensor Node can authenticate received public keys of both servers and generate two secrets and session keys to securely communicate with two servers independently. Finally, to authenticate the Sensor Node it encrypts its root secret using generated session key of Application Server and sends it to Application Server through the Network server. Application Server Authenticate the Sensor Node by comparing the root secret and inform Network Server with the authentication status.



P1 - ECC private key of End Node
g1 - ECC public key of End Node

P2 - ECC private key of Network server
g2 - ECC public key of Network server

P3 - ECC private key of Application Server
g3 - ECC public key of Application Server

root_secret_S - Root secret of sensor node
pA - Root private ECC key of Application Server
gA - Root public ECC key of Application Server

S1 - Shared secret between Node and Network server
S2 - Shared secret between Node and Application server

SK1 - Session key between Node and Network server

Figure 20: Initial key exchange with party authentications

4.5.3 Key Rolling

Key rolling can be done in two ways with this system. One is using the existing session key to encrypt a new session key and share it with other parties. This should be done before someone breach the existing session key. After a suspect situation of attack, we cannot trust the existing key anymore because of that rejoin is required. Sensor nodes are automated and they don't have resources to run any intelligence programs with them to detect suspicious situations. Hence, Network server should initiate a key exchange process. If Network servers ask Sensor node to initiate join, it creates some communication overhead on Sensor node and more communication steps need more power. And if the sensor node initiates the rejoin it should wait until the server

responses to complete the session key generation. In our solution, instead of asking the Sensor Node to initiate Network Server can initiate the rejoin as describe the steps in **Figure 21**.

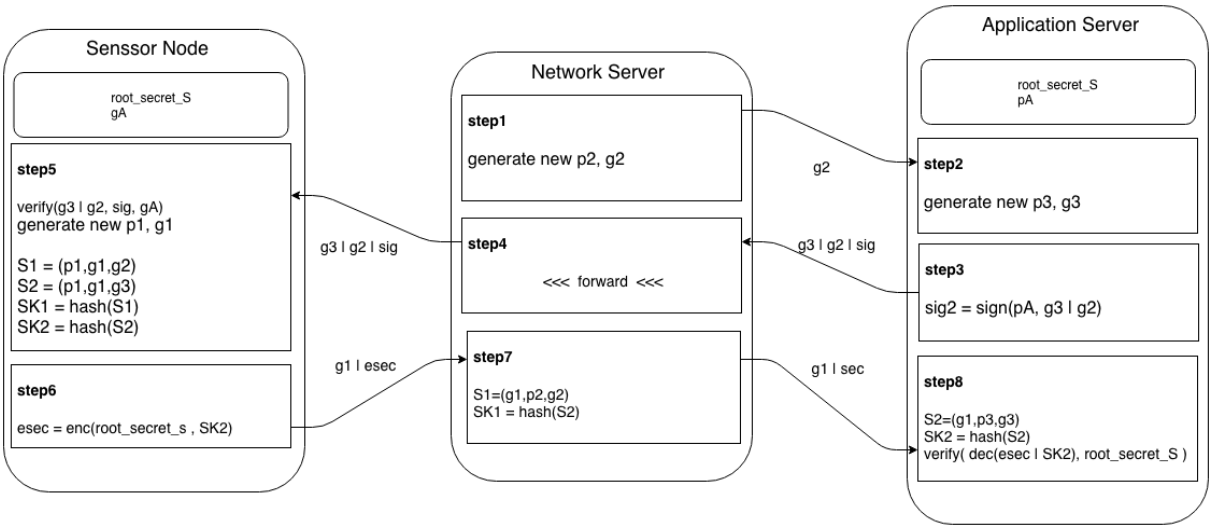


Figure 21: Re-join of sensor node initiate by Network Server.

Network server initiates the re-join of a Sensor Node by generating an ECC key pair for him and forward public key to the Application Server to create an authenticated message. The application server also generates a key pair and send the two public keys and the signature of them with Applications root private key. Then the Sensor Node receives join request with other parties' public keys. The sensor node can check the authentication of public keys and generate the session keys for both parties. Then Sensor Node can encrypt its root secret by new session key and send it with the public key to severs. The application can verify the Sensor Node by decrypting and checking the secret and both servers can generate the session keys.

4.6 Proof of concept implementation for Sensor Node

After identifying the appropriate implementation, key sizes and designed message flow we need an experimental design to test the capability of running the ECDH key exchange mechanism at the Sensor Node. When selecting implementation and at the evaluation of it we show that with existing LoRaWAN stack and implementation for two key exchange it remains very limited flash space(6% of total flash memory) for Application Code. Hence, we use general LoRa communication stack

for our experiment instead of heavy LoRaWAN stack. We design the experiment setup with two sensor nodes and a general-purpose computer as shown in **Figure 22**. Both Sensor Nodes have the same code with variables and parameters which needs for key exchange between three parties. We measure the flash memory, memory allocation for global data, stack memory allocation at run time and time consumption for cryptographic operations in key exchange mechanism at Sensor Node. We define variables for key exchange between three parties according to the proposed design. We execute cryptographic functions according to that design. But we don't implement Application server or Network Server sides. Instead of two servers, we use another LoRa node simulate the messages for three party key exchange. For a symmetric key generation, we use a simple mechanism which is selecting the first 128 bits out of 256-bit shared secret instead of using the hash function. This is not a security problem because 256 ECDH also provides 128-bit security level. We discuss the obtained results in the evaluation chapter.

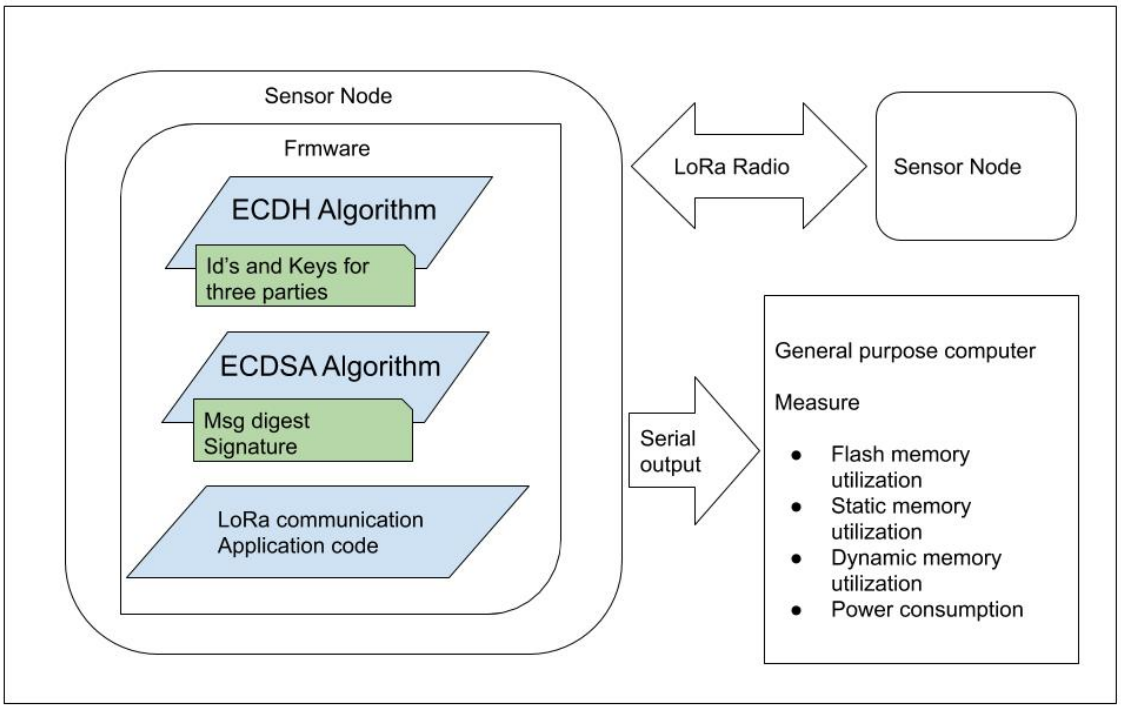


Figure 22: ECDH and ECDSA algorithms on Sensor node

4.7 summary

We identified the relations between identified variables for the key exchange mechanism for LoRaWAN. Before suggesting a solution, we set up the LoRaWAN system and investigate the resource availability of the Sensor Node. Then we discuss the selection of ECC implementation for LoRaWAN using existing literature and our experiments. Then we discuss the experiments to select a suitable key size for the key exchange. After that, we propose the basic key exchange mechanism for LoRaWAN. At the next part, we discuss the End party authentication of the key distribution and suggest the detailed design with a minimum number of message parsing for the key exchange. Finally, we discuss the re-joining of a Sensor Node to the system and proposed the steps to re-join with the minimum number of steps at the Sensor Node side for reducing the power consumption.

Chapter 5

Results and Evaluation

5.1 ECC Algorithm Evaluation

5.1.1 ECC implementation evaluation

Before selecting the ECC implementation for LoRaWAN we set up the LoRaWAN network and investigate the resource availability at the Sensor Node with minimum features of LoRaWAN stack as discussed in the previous chapter.

Features					Memory usage (Bytes)	
Debug Level	Interrupts	Ping	Beacon	OTAA	Flash Memory	Global Data
0	X	X	X	X	16370	1196
0	X	X	X	√	18532	1286

Table 3: Memory consumption with and without OTAA.

According to the results measured in **Table 3**, OTAA needs 2162 (18532-16370 = 2162) bytes out of flash memory size 28672 Bytes, which is approximately 7.54% flash memory. It uses 90 bytes which is 3.6 % of main memory size 2.5 KB. This statistic says that OTAA joining mechanism consumes relatively fewer resources at End Node. We also measure the main memory consumption of OTAA joining mechanism and observed main memory consumption at different stages as in Table 4. We observed maximum SRAM allocation at the data sending stage after Node joined to the Network. We also measure the time consumption for the OTAA joining procedure. It takes 5500 milliseconds on average with transmission times.

Status	SRAM[3] Allocation (Bytes)
Init join	1333
Send join request	1363
Joined	1345
Send Data	1381

Table 4: RAM allocation at LoRaWAN OTAA

In the previous chapter under ECC implementation selection, we discuss four implementations of Elliptic Curve for 8-bit AVR microcontrollers. Liu, Zhe, et al. have analysed and evaluated the resource consumptions of the other two implementations µKummer and Curve 25529 with their implementation FourQ. We want to evaluate ECC implementation with both key exchange and digital signature functionalities for our work. However, at Liu, Zhe, et al. evaluation digital signature algorithm is not there for Curve 25519. Hence, we take the evaluation results of µKummer and FourQ from Liu, Zhe, et al. work and we did experiments as describe in the previous chapter for Curve 25519 and SECP curves of Micro-ecc implementation. All the curves evaluate here provide 128-bit security strength.

Implementation	Function	Parameters	Memory	
			Code + data	stack
µKummer	ECDH	-	> 9,490	812
	ECDSA	-	> 16,516	992
FourQ / SchnorrQ	ECDH	$w^4 = 4$, $v^5 = 4$	30,820 + 980	2601
		w = 5, v = 5	35,484 + 980	2601
	ECDSA	$wp^6 = 6$	38,334 + 858	4957
		wp = 8	56,678 + 858	4957

[3] Static Random-Access Memory which uses flip-flops to store bits.
[4] W: window size of ECC scalar multiplication
[5] V: number of internal tables of ECC scalar multiplication
[6] wp:

Curve 25519	ECDH	-	11,408 + 468	512
	ECDSA	-	24,890 + 494	570
Micro-Ecc Secp256r1	ECDH	ol[7] = 2	11502+559	646
	ECDSA	ol = 2	14108+528	710
Micro-Ecc Secp256k1	ECDH	ol = 2	11578+559	646
	ECDSA	ol = 2	14184+527	710

Table 5: Memory consumptions for different ECC implementations for 8-bit AVR

According to Table 4, 12302 bytes of free flash memory space and 1364 of free RAM space is available with minimum features (without OTAA joining feature) of LoRaWAN stack. FourQ cannot run in any way in LoRaWAN Sensor node because required flash and RAM is very high than total space of the Sensor Node. To run the ECDSA algorithm with required variables it needs more than the available free space at the Sensor Node.

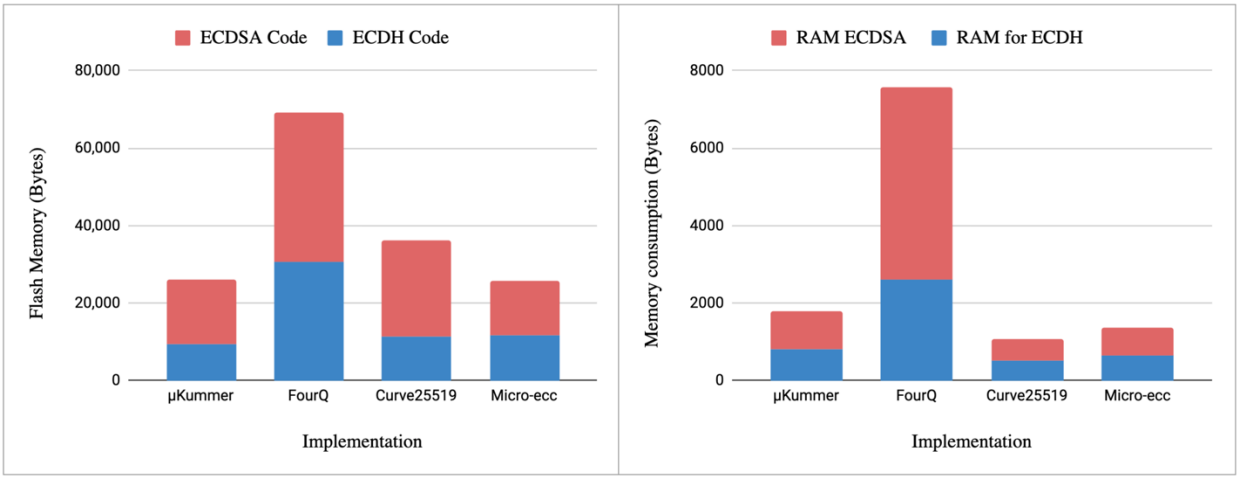


Figure 23: Flash Memory and Main Memory allocation of ECC implementation

Stack graphs in **Figure 23** show the total of flash memory allocations and main memory allocations for ECDH and EDSA for each implementation. Curve 25519 consumes lesser main memory but needs higher flash memory than free. And it is more than 80% of total flash memory for Curve25519 ECDSA. The µKummer and micro-ecc consume relatively less space

[7] ol - compiler optimization level for avr gcc

and Micro-ecc is the lesser. However, only ECDSA of µKummer consumes than 50% of Sensor node flash space and for micro-ecc it is less than 50%. An open source implementation of µKummer 8-bit AVR is also not available. Hence, we select Micro-ecc as the suitable implementation. Furthermore, to test the capability of using Micro-ecc. For that, we test with both ECDH and ECDSA with in the same code and obtained results in Table 7. It consumes less than flash memory usage for µKummer ECDSA and this verifies that Micro-ecc is more suitable than µKummer in memory wise.

Implementation	function	parameters	code+data	ram
Micro-Ecc Secp256r1	ECDH, ECDSA	ol = 2	14702 + 655	841
Micro-Ecc Secp256k1	ECDH, ECDSA	ol = 2	14778 + 655	841

Table 6: Memory consumption for both ECDH and ECDSA for Micro-ecc

5.1.2 Micro-ecc key sizes evaluation

As we discussed in previous chapter, after selecting the Micro-ecc implementation we do experiments to select a suitable key size. The experiment results are evaluated here in memory wise, efficiency wise and security level wise.

5.1.2.1 Memory allocation

According to the experimental setup of key size evaluation that we discussed in the previous chapter, we obtained results and here we graph memory them as visualized in **Figure 24**. It shows the memory allocation out of total space for both flash and RAM spaces. See Appendix 2 for the obtained results. Memory wise, we cannot observe a significant difference for different curves which represents different key sizes.

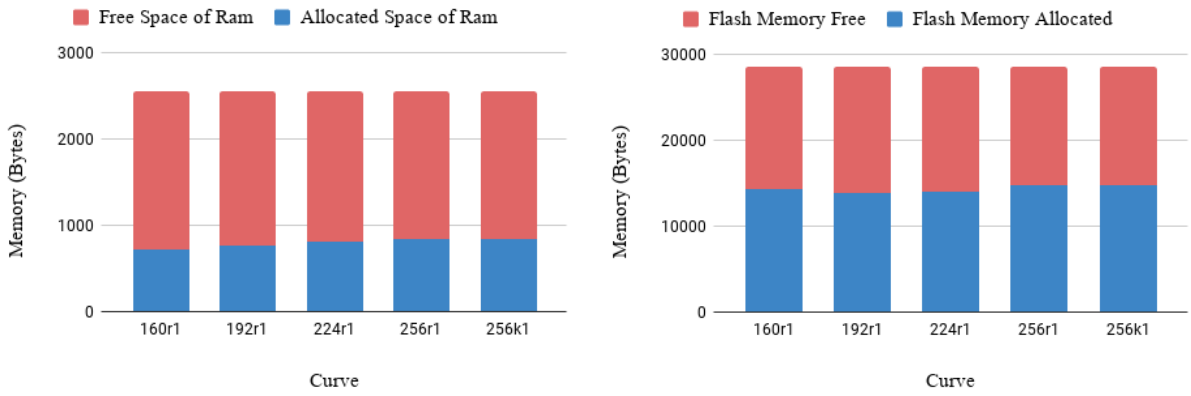


Figure 24: Flash memory and RAM consumption for different curves of Micro-ecc.

5.1.2.2 Time consumption

We measure the time consumption for different cryptographic functions of ECC with the same experimental setup for key size selection. Time consumptions are obtained for secret generation, key generation, signature generation and signature verification. Original results can see in Appendix 3 and here **Table 7** shows the Average values for time consumption of each function.

Curve	Key gen (ms)	Secret gen (ms)	signature gen (ms)	signature verify (ms)	total
160r1	2162	1996	2607	2299	9064
192r1	3322	3321	3769	3851	14263
224r1	5030	5031	5608	5877	21546
256r1	8236	8238	9021	9512	35007
256k1	6870	6871	7555	7577	28873

Table 7: Time consumption for both ECDH and ECDSA for Micro-ecc.

We visualized these results as a stack graph in **Figure 25**. According to the graph, short key sizes are more efficient than the higher key sizes. Higher key sizes provide higher security level than lower keys. Micro-ecc implementation provides two curves for 256 key size. 256r1 which known as random curve and 256k1 which known as Koblitz curve. In performance wise, Koblitz curve is better in all cryptographic functions.

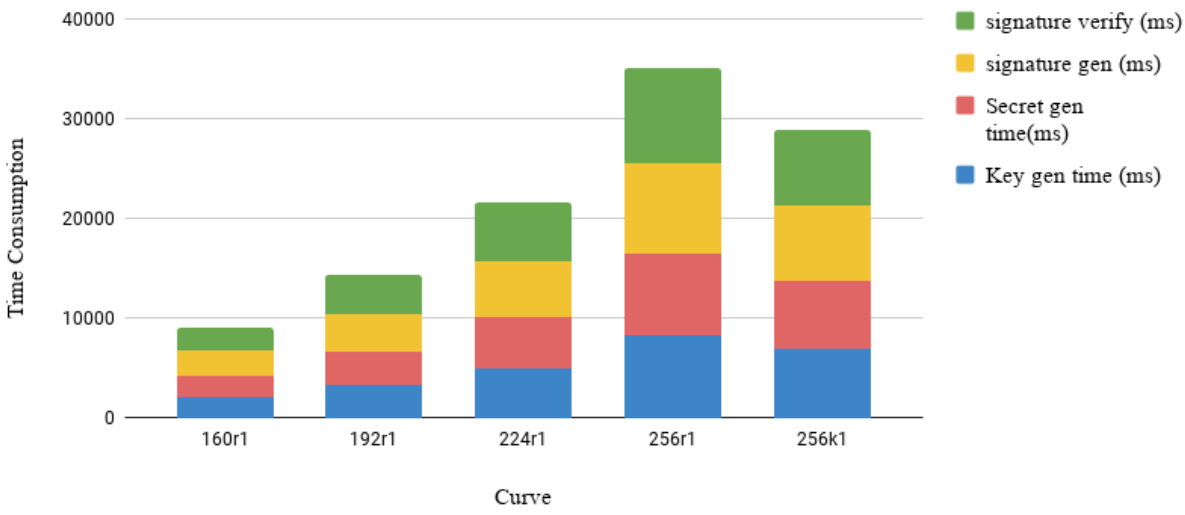


Figure 25: Performances for different curves of Micro-ecc.

5.1.2.3 Safety Level

We evaluate the key sizes in memory consumption wise and efficiency wise. Now we want to evaluate security wise. D. J. Bernstein and T. Lange have examined the safety of different elliptic curves [30]. They have discussed the safeness of Elliptic Curves against 11 security requirements. Curves in micro-ecc doesn't satisfy all 11 security requirements. Both secp256k1, secp256r1 doesn't satisfy 4 security requirements out of 11 and secp224r1 doesn't satisfy 5 security requirements. However, they provide a sufficient security level. The most popular cryptocurrency platforms Bitcoin [31] and Ethereum [32] also still uses secp256k1 curve because of its efficiency and safety. Hence here we propose to use secp256k1 standard curve for LoRaWAN End node. The security can be increased by doing the key exchange periodically. If so then we can also move to secp224r1 which is more resource efficient with the security level equals to 112-bit symmetric key.

5.2 Key Distribution Mechanism Evaluation

According to our experiment design for proof of concept key distribution in previous chapter we do the key exchange between Two LoRa nodes as the proof of concept and we includes all parameters and call number of executions of each cryptographic functions which needs to exchange keys between three parties as our Key exchange solution design in previous chapter. LoRaWAN Stack with basic functions of micro-ecc cryptography implementation consumes 94% of flash memory. Normal LoRa communication stack with required variables and functions for three party key exchange consumes 72% of Flash Memory and it consumes 1050 bytes of memory for global data and 1229 bytes of total RAM space allocation of the sensor node. Total RAM allocation is 48% at the Sensor Node. According to key exchange design it needs one key pair generation, two secret generation, one ECDSA verification and one AES encryption. According to Micro-ecc evaluation results we can calculate the time consumption for ECDH cryptographic functions.

$$cryptoTime = keyGeneration + 2 * secretGeneration + signature\ verification + AESencryptio$$
$$6870 + (6871 * 2) + 7577 = 28189\ ms$$

According to this calculation it takes 28.2 seconds which means almost half a minute for ECDH cryptographic calculations. This time is without two communication steps of proposed key exchange message flow from sensor node and network delays. OTAA consumes 5.5 second for whole joining procedure. Our key exchange consumes more than 6 times time compared to OTAA. But security wise this key exchange is more secure than OTAA.

We also want to discuss the security strength of our selected secret According to Wikipedia fastest supercomputer's performance is 143.5 PFLOPS[8] in 2018 [34] and according M. Arora's calculation [35] $7.5*10^{16}$ years needs a supercomputer to bruteforce the 128-bit secret. According to him, even 7 billion people in the world have super computers, this calculation takes millions of years. However, key can be revealed without brute forcing the symmetric keys. Breaking secp256k1 as D. J. Bernsteinn et al. shows or even a random try for guessing the symmetric key can be a success. But these possibilities are very rare. Hence, we can consider key rolling to increase the system security.

[8] The performance of a supercomputer is commonly measured in floating-point operations per second (FLOPS) and P in FLOPS means penta(10^{15})

ECDH key exchange with ECDSA is more secure than OTAA re-joining. Hence, it doesn't need frequent key rolling or re key exchange.

According to ATmega32u4 datasheet [15] Active microcontroller consumes 4.7mA with 8MHz clock speed, 3.6v and at $25C^0$. This voltage and speed are same as the specification of LoRaWAN Sensor Node. According to **Figure 21** it needs **two communications** for a node to do the key exchange for re-joining. Max message length is length of "pubKey1|pubKey2|signature" which is 64+64+64=192 Bytes. Table 1 of E. Aras et. al [6] shows 122 Bytes per second data rate for SF10 spread factor which also supports high distance communication. Then time for message transmission is 192/122 = 1.57 seconds. If we assume a maximum of 2 seconds per message, then it takes less than 4 second communication time for two communication messages of new re-joining. In this section we calculate crypto-time which is 28.2 seconds. Now the total calculated re-joining time is 28.2 + 4 = 32.2 seconds. With the network delays, we can take a maximum of 60 seconds which is almost double of calculated time for a re-join.

If user decide to do a key distribution for each day and with maximally **one minute communication time** with network delays, node consumes 4.7mA*(1*60/3600) h which is 0.07834mAh of battery capacity per day. Per a year it is 0.079mAh * 3600 which is 28.6 mAh and if we use a 3.7v, 4800mAh Li-Ion single battery to power the node, it consumes 28.6/4800 mAh which is 0.6% of total capacity from the battery. This is bearable and less capacity consumption percentage with 4800mAh Li-Ion single battery which is available in local market.

5.3 Summary

We discuss experiment designs and proposed solution at the previous chapter and this chapter evaluate the results obtained at each experiment. First, we discuss LoRaWAN resource consumption results. Then we evaluate ECC implementation results and key size selection results. Finally, we discuss the strengthens and weaknesses of the proposed solution. Under the evaluation of proposed key exchange, we discuss the ability to run it on the sensor node with its memory constraints. Then we discuss the security strengths and requirement of rejoining. At the end, we discuss power consumption and overhead for the Sensor nodes battery life.

Chapter 6

Conclusion and Future Works

We studied the Current status of IOT and the emergence of Low Power Wide Area Networks. Then we focus the LoRaWAN which is the most widely spread LPWAN in the world. Hence Security is the main concern of each system we studied the LoRaWAN Security. Our literature review highlights the security vulnerabilities End Node Device joining mechanism of the LoRaWAN. We Discussed the existing joining mechanism and proposed solutions to increase the security of Sensor Device Joining. Few researchers have raised the need of a key exchange mechanism and we found one ECDH base proposal solution using existing literature. We identified the gap of not focusing the Sensor Node Device limitations when integrating ECDH for LoRaWAN.

Our work identifies the capability of running ECDH based key exchange at LoRaWAN End Node Devices. We do our research limiting to specifications of exact End Node of LoRaWAN system. We found an ECC implementation which can perform with the Node limitations. Evaluating resource requirements of four implementations of ECC we select Micro-ecc implementation for LoRaWAN Key exchange. We also evaluate different curves support by Micro-ecc against memory consumption, efficiency and security strengths and select secp256k1 standard Koblitz curve which uses 256 keys size and provide 128-bit level security strength. We proposed a key distribution mechanism with minimum message parsing to save the Sensor Node battery power. This solution exchange two secrets to communicate between Network Server and Application server from the Sensor Node. We use a root secret at End Node side and root ECC key pair at Application server side to provide the authentication for the key exchange. See all the source codes we use for testing and the source code for proof of concept at the GitHub [43].

Our work only implements a proof of work to prove the capability of running micro-ecc based key exchange for LoRaWAN Sensor Node. For future works, it needs to implement the full system

with the ECDH key exchange. Flash memory limitation is the main constraint we found for ECDH implementation at sensor Node. We see more efficient and secure ECC implementations which cannot perform on LoRaWAN Node because of memory limitations. For future work, we suggest doing research to select the most suitable microcontroller for Low Power Wide Area Networks. In this case, it needs to consider more flash memory and ram availability and less power consumption. Cost of the microcontroller also should consider. In our research, we use only the existing implementations of ECC and LoRaWAN stack. Some of ECC implementation is not available for 8-bit AVR microcontrollers. Implement a more resource efficient, power efficient and secure ECC implementation for 8-bit AVRs is also a need the sensor network area.

References:

[1] A. Augustin, J. Yi, T. Clausen, and W. Townsley, “A Study of LoRa: Long Range & Low Power Networks for the Internet of Things,” Sensors, vol. 16, no. 9, p. 1466, Sep. 2016.

[2] "LoRaWAN® Specification v1.0.2 | LoRa Alliance™", Lora-alliance.org, 2019. [Online]. Available: https://lora-alliance.org/resource-hub/lorawantm-specification-v102. [Accessed: 19- Sep- 2019].

[3] "LoRaWAN® Back-End Interfaces v1.0 | LoRa Alliance™", Lora-alliance.org, 2019. [Online]. Available: https://lora-alliance.org/resource-hub/lorawantm-back-end-interfaces-v10. [Accessed: 19- Sep- 2019].

[4] "LoRaWAN® Specification v1.1 | LoRa Alliance™", Lora-alliance.org, 2019. [Online]. Available: https://lora-alliance.org/resource-hub/lorawantm-specification-v11. [Accessed: 19- Sep- 2019].

[5] X. Yang, E. Karampatzakis, C. Doerr, and F. Kuipers, “Security Vulnerabilities in LoRaWAN,” 2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI), 2018.

[6] E. Aras, G. S. Ramachandran, P. Lawrence, and D. Hughes, “Exploring the Security Vulnerabilities of LoRa,” 2017 3rd IEEE International Conference on Cybernetics (CYBCONF), 2017.

[7] K. Feichtinger, Y. Nakano, K. Fukushima, and S. Kiyomoto, “Enhancing the Security of Over-The-Air-Activation of LoRaWAN Using a Hybrid Cryptosystem,”International Journal of Computer Science and Network Security, vol. 18, no. 2, pp. 1–9, Feb. 2018.

[8] M. V. Leent, “An improved key distribution and updating mechanism for low power wide area networks (LPWAN),”, Openaccess, 2017.

[9] "The Things Network", Thethingsnetwork.org, 2019. [Online]. Available: https://www.thethingsnetwork.org/. [Accessed: 21- Sep- 2019].

[10] Mekki, K., Bajic, E., Chaxel, F. and Meyer, F. (2019). A comparative study of LPWAN technologies for large-scale IoT deployment. ICT Express, 5(1), pp.1-7.

[11] Sung, Woo-Jin, Hyeong-Geun Ahn, Jong-Beom Kim, and Seong-Gon Choi. "Protecting end-device from replay attack on LoRaWAN." In Advanced Communication Technology (ICACT), 2018 20th International Conference on, pp. 167-171. IEEE, 2018.

[12] Tomasin, Stefano, Simone Zulian, and Lorenzo Vangelista. "Security analysis of lorawan join procedure for internet of things networks." In Wireless Communications and Networking Conference Workshops (WCNCW), 2017 IEEE, pp. 1-6. IEEE, 2017.

[13] "The Things Uno", The Things Network, 2019. [Online]. Available: https://www.thethingsnetwork.org/docs/devices/uno. [Accessed: 21- Sep- 2019].

[14] "The Things Node", The Things Network, 2019. [Online]. Available: https://www.thethingsnetwork.org/docs/devices/node. [Accessed: 21- Sep- 2019].

[15] Atmel, “Low-Power Long Range LoRa® Technology Transceiver Module,” Atmel-7766J-USB-ATmega16U4/32U4-Datasheet, 2015. [Online]. Available: http://ww1.microchip.com/downloads/en/devicedoc/atmel-7766-8-bit-avr-atmega16u4-32u4_datasheet.pdf. [Accessed: 19- Sep- 2019].

[16] Microchip, “8-bit Microcontroller with 16/32K bytes of ISP Flash and USB Controller,” DS50002346C Datasheet, April. 2016. [Online]. Available:

http://ww1.microchip.com/downloads/en/devicedoc/atmel-7766-8-bit-avr-atmega16u4-32u4_datasheet.pdf. [Accessed: 19- Sep- 2019].

[17] Semtech, “SX1276/77/78 - 137-1050 MHz Ultra Low Power Long Range Transceiver,” Datasheet, Jul. 2012. [Online]. Available: https://upverter.com/datasheet/b31dec1362ed0746786e1d3a0bf1fe1ff5f6eb0c.pdf. [Accessed: 19- Sep- 2019].

[18] BSFrance, “LoRa32u4II- Low power Atmega® 32u4 LoRa 868Mhz 915Mhz compact board with 3.7V LiPo cell management,” Datasheet, 2017. [Online]. Available: https://docs.bsfrance.fr/documentation/11355_LORA32U4II/Datasheet_LoRa32u4II_1.1.pdf. [Accessed: 19- Sep- 2019].

[19] "matthijskooijman/arduino-lmic", GitHub, 2019. [Online]. Available: https://github.com/matthijskooijman/arduino-lmic. [Accessed: 19- Sep- 2019].

[20] Z. Liu, P. Longa, G. Pereira, O. Reparaz and H. Seo, "FourQ on embedded devices with strong countermeasures against side-channel attacks", IEEE Transactions on Dependable and Secure Computing, pp. 1-1, 2018. Available: 10.1109/tdsc.2018.2799844.

[21] "Elliptic Curve Cryptography - OpenSSLWiki", Wiki.openssl.org, 2019. [Online]. Available: https://wiki.openssl.org/index.php/Elliptic_Curve_Cryptography. [Accessed: 19- Sep- 2019].

[22] "micro-ecc", Arduinolibraries.info, 2019. [Online]. Available: https://www.arduinolibraries.info/libraries/micro-ecc. [Accessed: 19- Sep- 2019].

[23] "Arduino Cryptography Library: Curve25519 Class Reference", Rweather.github.io, 2019. [Online]. Available: https://rweather.github.io/arduinolibs/classCurve25519.html. [Accessed: 19- Sep- 2019].

[24] "Elliptic-curve Diffie–Hellman", En.wikipedia.org, 2019. [Online]. Available: https://en.wikipedia.org/wiki/Elliptic-curve_Diffie–Hellman. [Accessed: 19- Sep- 2019].

[25] "The Things Gateway", The Things Network, 2019. [Online]. Available: https://www.thethingsnetwork.org/docs/gateways/gateway/. [Accessed: 19- Sep- 2019].

[26] ESP8266EX Datasheet, V 6.2. Espressif, 2019. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf. [Accessed: 19- Sep- 2019].

[27] RFM95/96/97/98(W) - Low Power Long Range Transceiver Module V1.0. HOPE MICROELECTRONICS CO.,LTD, Shenzhen, Chain. 2006. [Online]. Available:https://cdn.sparkfun.com/assets/learn_tutorials/8/0/4/RFM95_96_97_98W.pdf. [Accessed: 19- Sep- 2019].

[28] IBM, Matthis Kooijman, Terry Moore, ChaeHee Won, Frank Rose, “MCCI LoRaWAN LMIC library,” mcci-catena/arduino-lmic. IBM. [Online]. Available: https://github.com/mcci-catena/arduino-lmic. [Accessed: 19- Sep- 2019].

[29] Certicom Research, “SEC 2: Recommended Elliptic Curve Domain Parameters.” Certicom Corp., 09-Sep-2000 [Online]. Available : https://www.secg.org/SEC2-Ver-1.0.pdf. [Accessed: 19- Sep- 2019].

[30] D. J. Bernstein and T. Lange, "Safe Curves: choosing safe curves for elliptic-curve cryptography", Safecurves.cr.yp.to, 2019. [Online]. Available: http://safecurves.cr.yp.to/. [Accessed: 21- Sep- 2019].

[31] Nakamoto, Satoshi. "Bitcoin: A peer-to-peer electronic cash system." (2008).

[32] Wood, Gavin. "Ethereum: A secure decentralised generalised transaction ledger." Ethereum project yellow paper 151 (2014): 1-32.

[33] “Hardware Security Modules (HSMs) - SafeNet Encryption & Key Security,” Gemalto. [Online]. Available: https://safenet.gemalto.com/data-encryption/hardware-security-modules-hsms/. [Accessed: 05-Jun-2019].

[34] “Supercomputer,” Wikipedia, 09-May-2019. [Online]. Available: https://en.wikipedia.org/wiki/Supercomputer. [Accessed: 05-Jun-2019].

[35] M. Arora and Freescale Semiconductor, “How secure is AES against brute force attacks?,” EETimes, 07-May-2012. [Online]. Available: https://www.eetimes.com/document.asp?doc_id=1279619. [Accessed: 05-Jun-2019].

[36] “RSA (cryptosystem),” Wikipedia, 23-May-2019. [Online]. Available: https://en.wikipedia.org/wiki/RSA_(cryptosystem). [Accessed: 06-Jun-2019].

[37] Maniacbug, “maniacbug/MemoryFree,” GitHub, 24-Apr-2011. [Online]. Available: https://github.com/maniacbug/MemoryFree. [Accessed: 07-Jun-2019].

[38] “Diffie–Hellman key exchange,” Wikipedia, 30-May-2019. [Online]. Available: https://en.wikipedia.org/wiki/Diffie–Hellman_key_exchange. [Accessed: 07-Jun-2019].

[39] “Elliptic Curve Digital Signature Algorithm,” Wikipedia, 13-May-2019. [Online]. Available: https://en.wikipedia.org/wiki/Elliptic_Curve_Digital_Signature_Algorithm. [Accessed: 07-Jun-2019].

[40] IAR Embedded Workbench. [Online]. Available: https://www.iar.com/iar-embedded-workbench/. [Accessed: 07-Jun-2019].

[41] J. Renes, P. Schwabe, B. Smith and L. Batina, "µKummer: Efficient hyperelliptic signatures and key exchange on microcontrollers", in International Conference on Cryptographic Hardware and Embedded Systems, Springer, Berlin, Heidelberg, 2016, pp. PP 301- 320.

[42] iSECPartners, “iSECPartners/nano-ecc,” GitHub, 12-Jul-2013. [Online]. Available: https://github.com/iSECPartners/nano-ecc. [Accessed: 07- Jun- 2019].

[43] N. Jayasuriya, “NamalJayasuriya/ECDH-for-LoRaWAN,” GitHub, 07-Jun-2019. [Online]. Available: https://github.com/NamalJayasuriya/ECDH-for-LoRaWAN. [Accessed: 07- Sep -2019].

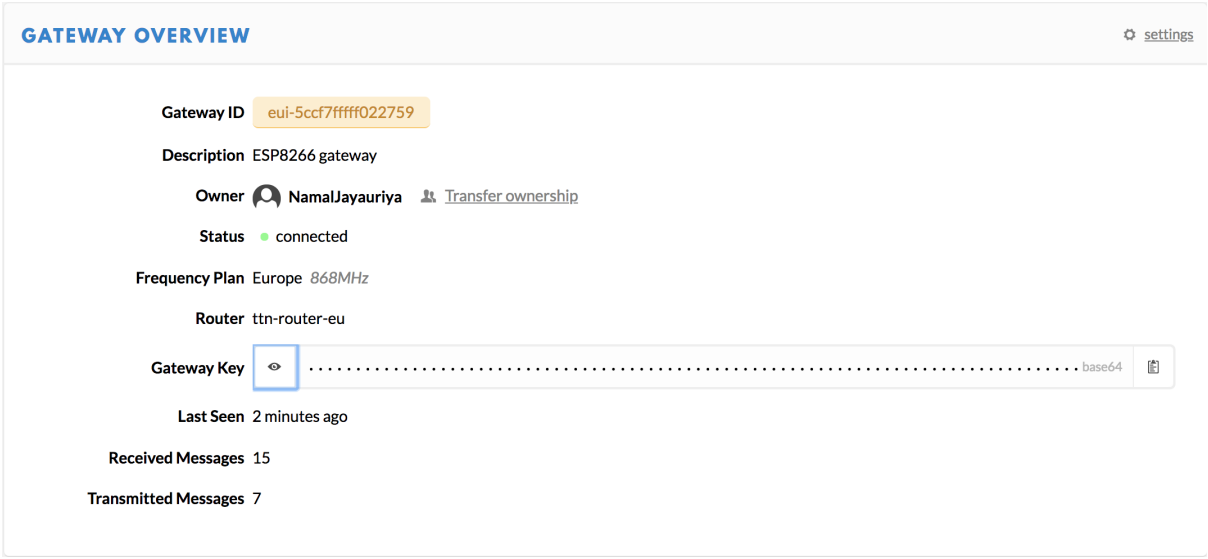
[44] “Arduino - Home,” Arduino - Introduction. [Online]. Available: https://www.arduino.cc/. [Accessed: 02-Dec-2018].

[45] ATmel, “8/16-bit XMEGA A3 Microcontroller.” Atmel Corporation, 2009. [Online] Available: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-8068-8-and16-bit-AVR-XMEGA-A3-Microcontrollers_Datasheet.pdf. [Accessed: 07- Sep -2019].

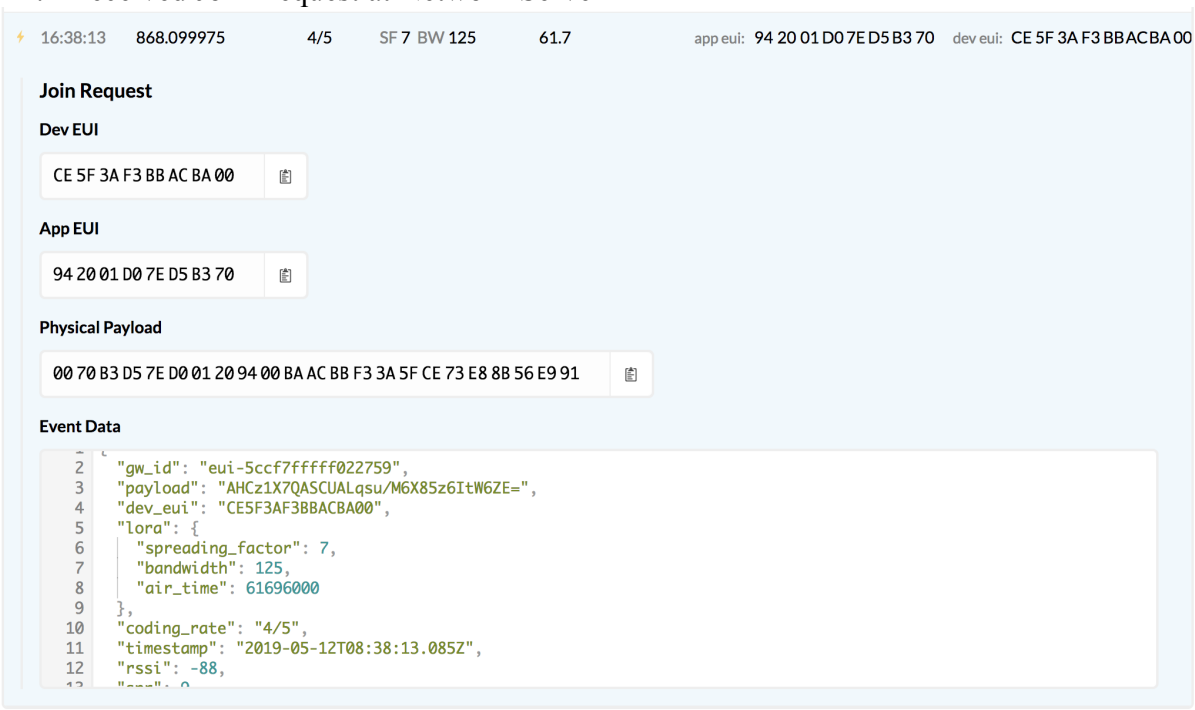
Appendix

Appendix 1

A1.1 Registered LoRaWAN gateway status at Network Server



A1.2 Received Join Request at Network Server



A1.3 Generated Join Accept message at Network Server

Appendix 2

A2.1 Memory allocation results for Micro-ecc different Key Sizes.

Curve	Flash Memory Allocated	Flash MEmory Free	Allocated Space of Ram	Free Space of Ram
160r1	14302	14370	723	1837
192r1	13828	14844	761	1799
224r1	14064	14608	801	1759
256r1	14702	13970	841	1719
256k1	14778	13894	841	1719

Appendix 3

A3.1 Time consumption results of cryptographic functions for micro-ecc 160r1

160r1			
keygen	secret gen	sign	verify
2364	1989	2357	2240
2268	1997	2906	2302
2087	1992	2450	2382
2001	1995	2531	2284
1996	1997	2446	2291
2177	1996	2628	2329
1994	1993	2808	2300
1996	1991	2441	2249

1996	1986	2454	2341
2363	1995	2459	2181
2890	1999	2529	2380
2001	1992	3275	2273
2089	1998	2716	2339
2107	1999	2374	2236
2005	1993	2552	2308
2191	2001	2577	2314
2109	2005	2378	2337
2005	2001	2377	2339
2429	2003	3093	2318
2175	1997	2785	2233
Averages :			
2162.15	1995.95	2606.8	2298.8

A3.2 Time consumption results of cryptographic functions for micro-ecc 192r1

192r1			
keygen	secret gen	sign	verify
3333	3313	3758	3861
3318	3314	3752	3926
3314	3314	3748	3889
3312	3312	3748	3826
3313	3307	3752	3943
3311	3308	3751	3765
3315	3318	3750	3832
3309	3316	3752	3825
3316	3307	3750	3701
3314	3322	3756	3797
3318	3330	3785	3774
3330	3328	3786	4018
3333	3336	3778	3914
3334	3333	3780	3914
3328	3328	3791	3873
3324	3326	3789	3809

3330	3328	3787	3848
3330	3330	3788	3859
3334	3332	3785	3877
3326	3328	3791	3787
Averages :			
3322.1	3321.5	3768.85	3851.9

A3.3 Time consumption results of cryptographic functions for micro-ecc 224r1

224r1			
keygen	secret gen	sign	verify
4986	5026	5595	5939
5020	5040	5599	5892
5026	5022	5593	5904
5036	5040	5630	5872
5046	5042	5597	5907
5030	5034	5607	5851
5034	5022	5601	5773
5026	5030	5612	5873
5024	5024	5603	5853
5030	5032	5601	5843
5030	5034	5620	5972
5046	5027	5604	5908
5038	5039	5634	5853
5046	5024	5593	6009
5028	5029	5600	5884
5029	5026	5634	5909
5032	5034	5615	5772
5032	5030	5606	5771
5028	5034	5599	5869
5030	5034	5626	5890
Averages :			
5029.85	5031.15	5608.45	5877.2

A3.4 Time consumption results of cryptographic functions for micro-ecc 156r1

256r1			
keygen	secret gen	sign	verify
8233	8231	9097	9484
8108	8215	8976	9573
8210	8210	8968	9552
8209	8221	9062	9329
8278	8286	9112	9701
8262	8283	8994	9528
8231	8243	9019	9693
8233	8221	8981	9400
8223	8266	9072	9718
8268	8267	9073	9748
8258	8211	8956	9478
8202	8206	8970	9288
8251	8262	9074	9653
8265	8211	8955	9647
8255	8223	9022	9322
8210	8211	8974	9523
8276	8283	9113	9323
8282	8223	8980	9603
8260	8263	9036	9505
8214	8215	8976	9278
Averages :			
8236.4	8237.55	9020.5	9517.3

A3.5 Time consumption results of cryptographic functions for micro-ecc 256k1

256k1			
keygen	secret gen	sign	verify
6848	6873	7547	7670
6873	6869	7555	7670
6873	6871	7555	7420
6865	6867	7559	7694

6873	6867	7560	7628
6878	6873	7551	7506
6877	6877	7563	7572
6875	6867	7557	7348
6865	6871	7549	7633
6873	6875	7551	7528
6869	6867	7555	7874
6875	6863	7557	7385
6869	6871	7559	7799
6873	6869	7557	7483
6867	6867	7557	7672
6871	6877	7555	7471
6866	6878	7549	7559
6867	6875	7551	7661
6869	6867	7562	7420
6871	6872	7551	7542
Averages :			
6869.85	6870.8	7555	7576.75