UNIVERSITY OF LIVERPOOL
DEPARTMENT OF ELECTRICAL ENGINEERING & ELECTRONICS

Final report for project 'LoRa-Based Key Generation in Low Power Wide Area Networks'

Author: Alexandru Mateescu (201500558)

Project Supervisor: Professor Zhang Junqing

Project Assessor: Professor Alan Marshall

Declaration of academic integrity

The standard University of Liverpool statement of academic integrity [6] should go here as follows:

I confirm that I have read and understood the University's Academic Integrity Policy.

I confirm that I have acted honestly, ethically and professionally in conduct leading to assessment for the programme of study.

I confirm that I have not copied material from another source nor committed plagiarism nor fabricated, falsified or embellished data when completing the attached piece of work. I confirm that I have not copied material from another source, nor colluded with any other student in the preparation and production of this work.

SIGNATURE……………………Alexandru…Mateescu…………..……………………. DATE………18/04/2022……….

Abstract

The report below discusses the implementation, industrial impact, design, and test results of a primarily a software final year project. In the project a common key used for cryptography is generated securely by a key generation algorithm. The key can be used to encrypt and decrypt data or plaintext for low-power IoT (Internet of Things) devices over long distances.

The IoT devices in question, are two evaluation boards from SEMTECH that wirelessly communicate with each other using the LoRa communication protocol.

This project assumes that by sampling and recording the signal strength of the distance between the two boards, a unique key can be generated.

To use this method to create a key, an algorithm is needed. This is called a key generation algorithm, and it involves four main steps that are required to obtain the commonly generated key and the resulting hash key.

1. Channel probing. A message is being sent between the boards and each board records the RSSI of the received message. These values are sampled 144 times and if any message packets are lost, they will be retransmitted by an original packet-matching algorithm.
2. Quantization. The RSSI values that the boards have received are transformed into binary values.
3. Information reconciliation. Here a unique algorithm has been developed similar to BCH code, although this algorithm is more versatile and complex.
4. Privacy amplification. A hash key is generated from the generated key. This hash key can be used as a signature or proof that two devices hold the same key without leaking the key itself.

Once this key is created, the secure wireless communication of the two wide-area IoT devices is assured. This is because data can now be encrypted and decrypted using the generated key.

Table of Contents

Project Specifications

Learning about Key Generation and LoRa

During the first three weeks of the academic year, thorough academic research is carried out to understand the overall concept behind the LoRa protocol. During this stage, various articles are read, and technical limitations are taken into consideration and noted. The key literature review is being analysed around the concepts of key generation, network protocols, physical and MAC layers, technical limitations and features, the key concept, and abbreviations in telecommunications. Approaching such abstract notions, the indirect implications of these work tasks sum together into the Theoretical Work package.

Key Generation and User Interface Work Package

Starting work on practical tasks represents entering an entirely new stage in the development process. This is the Practical Package (Or Key Generation and UI work package).

A. *Starting Practical Work*

Once lecturing is over, the next week (Week 4) will focus on wirelessly connecting two development boards with the use of the LoRaWAN protocol. A C++ algorithm will be used to check this stage. It is a Ping-Pong algorithm, where a message is sent from development board A to development board B with the message ‘Ping’. Once board B receives the message it will send back a ‘Pong’ message to board A. This task is meant to replicate the ACK (acknowledgement) message that will later be used for the imaginary IoT devices representing Alice and Bob.

B. *Channel Probing*

During weeks 7 and 8, a more complex algorithm will be implemented and tested between the two boards. This algorithm will fully replicate the Alice and Bob interaction by using the key encryption algorithm provided here [1]. This is the Channel Probing milestone.

C. *Quantization*

Once these tasks are carried out, during weeks 9 and 10, the boards will fully replicate the generated encryption protocol based on the randomness of the boards. The analog-to-digital conversion will be quantized and would produce a binary key almost identical for both devices. Almost identical because bit correction has not been implemented at this stage. This will allow a fully encrypted connection between Alice and Bob in a low-powered and wide area network.

D. *Information Reconciliation*

During week 2 of the second semester, further tests will be carried out, and bit correction for the divergences in bits between Alice and Bob will be implemented.

Improvements

Various Improvements During the holiday weeks of Christmas, the algorithm will be improved. At this stage, maintenance of the devices will be carried out and the algorithm will be updated for better performance. Tests with different devices in various situations will be made. Such as carrying the devices outside, checking the data rates at different distances, and adding more eavesdroppers. All this information will be noted and included in the final report alongside the results. Unforeseen events can happen and having a flexible schedule can save an entire project. This period will also be used in case the work project would be delayed for any reason.

Final Improvements At this stage, the project has reached its peak state. The algorithm is fully implemented, and tests validate its functionality of it. A third development board will be introduced during week 5 of the second semester. We will call the third device Ron. Ron is not a benevolent user and will try to compromise the data transfer between Alice and Bob. Ron will be the eavesdropper trying to steal the information being sent from Alice to Bob. It will also try to interfere with their data transfer from various distances.

Introduction

Through the course of your reading, please keep in mind that most algorithm implementations and designs are original and very few programming libraries have been used for the development of the key generation algorithm. The main target for this project was to implement as many unique adaptations and original designs as possible. Because the main target of a university student should be to learn, not to memorise. And the best way to learn is to create something unique. If the thesis of a project only describes pre-existing concepts and contains a disproportional amount of paraphrased content, then the project is not an original project, but just an intersection of borrowed ideas. So before jumping into comments because you the reader are slightly confused, understand that this is project is experimental and original, something much harder to achieve than merging libraries and research. With this in mind, please, enjoy your reading.

Security can often be an issue as IoT devices continue to make their presence more often in the quotidian world. Having a long-range communication protocol, such as LoRa, that is reliable and open source, means that developers can easily implement and expand it.

In the sections below, the key concepts of this project will be briefly explained. Namely what Key Generation and LoRa are, and how these two concepts interact and merge to form the thesis of this project.

The below sub headers will explain the presented concepts in more broad terms. More in depth and project specific topics will be touched on later in the report. The technical terms and concepts below are presented to familiarise the reader with the project's key notions.

What is key generation?

This introduction does not have the purpose of explaining how key generation works for this project in detail. It is here to explain how Key Generation works as a general concept, so that you, the reader, can understand the key concepts before diving into the essential technical functionalities of the project later in the report.

One of the recently developed technology that ensures long-range communication for IoT devices is successful and secure is called Key Generation.

The idea of cryptography and the use of private keys to encrypt data has been present for thousands of years. However, the modern concept of key generation, which involves generating and distributing keys for use in encryption and decryption, emerged in the 1970s with the development of public key cryptography by Whitfield Diffie and Martin Hellman. [2]

The main concept of Key Generation is to generate a hidden or private key that can be used for encryption. With encryption, unencrypted messages in human-readable form, or plaintext can be converted into a format that is unreadable without the decryption key.

With the use of mathematical permutations, a unique key can be generated for encrypting and decrypting data. The key size should be large and random to prevent malicious users from intercepting or modifying the message with the use of brute force attacks. [3]

In this project, key generation creates a local private and common key for Alice and Bob. This method is different and less complex from the widely used public key cryptography which can also be created by using key generation.

The key generation algorithm for this specific project involves four main steps which will be discussed further in the coming chapters. Namely Channel Probing (with packet matching included), Quantization (which can have different and various quantization algorithms and implementations), Information Reconciliation (this can also use different algorithms) and Privacy Amplification (which can use different accepted encoding methods by NIST [4], ENISA [5], IETF [6], etc. for generating a hash key that can be of different lengths depending on the coding method used, i.e. 128, 256 bits…).

The main idea of key generation is to ensure that the generated keys used for encryption and decryption are secure and secret. And that they can be generated in a way that is too difficult, wastes too many resources, or is too complex or time-consuming to deduce or guess by attackers.

What is LoRa?

LoRa (Long Range) is a low-power, long range communication protocol used for Internet of Things applications. It is widely used for long-range data transmission between IoT devices with low power consumption and limited data rates. [7]

LoRa uses Chip Spread Spectrum (CSS) which is a modulation technique that permits the signal to be sent over long ranges using a minor amount of power. [8] This means that data can be sent over distances that can range from 5km in urban areas where are various electromagnetic interferences to 15km in rural areas or in open fields. [9]

LoRa Alliance is a non-profit organisation that promotes development and reverse engineered the LoRaWAN protocol. LoRa is an open-source software and hardware technology, making it widely available to the general public to contribute to its development.

Semtech manufactures most microchips and boards that use the LoRa protocol. The SX1280 board used for this project is developed by Semtech, and the NUCLEO-L073RZ microcontroller is developed by STM, so an MBed compatible platform can be used to implement the code.

Altogether, LoRa offers a low-power, wide-area wireless communication suitable for IoT applications that require long-range connectivity and low power consumption.

Key generation and LoRa

As mentioned in the key generation header of the introduction, for this project a common key for the two IoT devices is being generated where the used communication protocol to achieve this is LoRa.

Below is detailed a broad outlook of the chosen implementation. The LoRa protocol is following a similar working pattern to other IoT secure communication technologies such as LoRaWAN or Public Key Encryption.

In general terms, there are four major stages to generate the keys as shown in the Figure 1. These stages are the same for LoRaWAN, however below is specified why and how these factors work for LoRa.

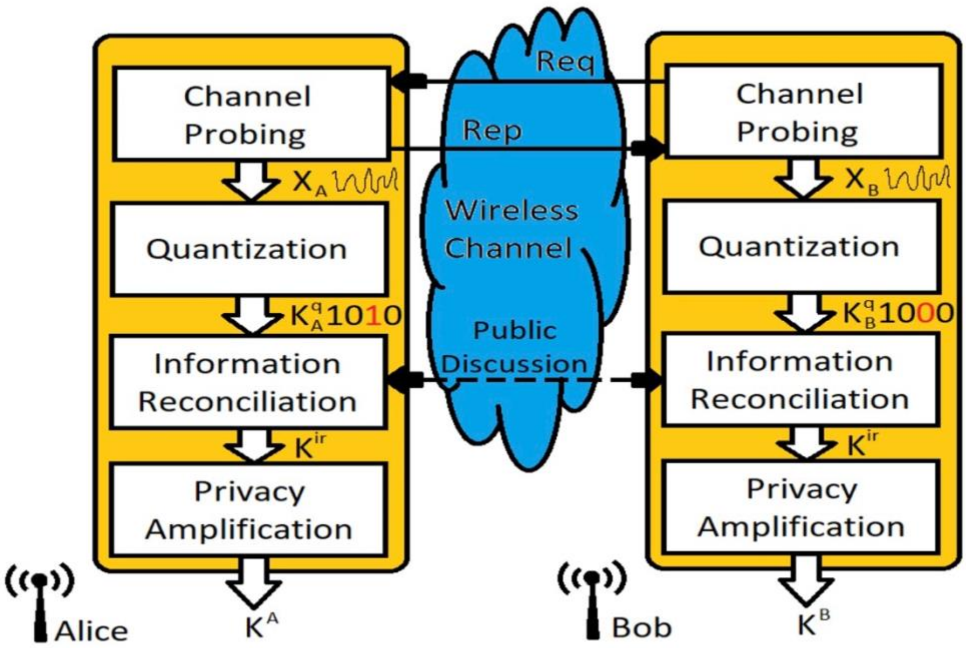


Figure 1 - Key Generation Process

1) Pre-key generation.
   This stage can be further broken down into channel probing, measurement pre-selection, measurement match and pre-correction. [10] Here a unique 'pre-key' is locally generated and secretly stored in each device of the network. In this case in the LoRa network there are only two devices, namely Alice and Bob, or Master and Slave. The pre-keys are generated by sampling the sent message RSSI (although it could also use the SNR) and recording privately. In a dynamic environment a semi-randomised or unpredictable key can be derived. Presumably this key should be too resource consuming to decipher by a third malicious user. The pre-keys are stored securely on each device and are used to derive the root keys [11]. For measurement matching and pre-correction, a packet matching algorithm has been implemented where the number of received packets is recorded at Bob's side and sent to Alice. Bob only sends a message after it has received a

message from Alice. Alice discards any messages which do not locally match the counted packet value of Bob's message.

2) Root key generation.
   In this step, the root keys are derived from the pre-keys using a key deviation function. For this project a uniform quantization algorithm is used to derive the root keys (which are the master and slave keys, or K_A and K_B) from the pre-key (which is the vector of RSSI values). [12]

3) Session key generation.
   For this project, algorithm session keys are not used, this technique is mainly used in LoRaWAN and in public key encryption. [13] For this project there is no need to have a Server node, because both Alice and Bob "act" as both server and node. Therefor there is no need for a session key to validate the integrity of each message. LoRaWAN uses AES for the Message Integrity Code (MIC). For this project, the message integrity of the message Alice or Bob send is encoded using SHA-256. Like AES since both are 256 bits, however AES has greater performance. [14]

4) Key update.
   Periodically breaches in the system might appear, and updating the generated keys might be a requirement. It was not a necessity to implement this for the purpose of presenting the functionality and security of this project. [15]

Literature survey

In the research side of this project and for the early-stage implementations of the project, most used articles were the ones published by Junqing Zhang and Alan Marshall. Especially useful were the Differential Based Quantization and the Information Reconciliation Secure Sketch algorithms in the 'Channel-Envelope Differencing Eliminates Secret Key Correlation: LoRa-Based Key Generation in Low Power Wide Area Networks' paper from the IEEE Transactions on Vehicular Technology research journal. These algorithms helped implement the quantization code and provided much of the inspiration for the information reconciliation algorithm. These algorithms are cited and talked about later in the Design chapter of this report.

Industrial relevance, real-world applicability, and scientific/societal impact

Industrial impact

By 2024 it is expected that the IoT industry will generate a revenue of 4.3 trillion dollars [16] within the fields of manufacturing, infrastructure, connectivity, and other services. The value and number of IoT devices is appreciated to increase considerably more in the future.

Businesses that address the need for IoT technologies, include, but are not limited to smart cities, smart power, transport and logistics, agriculture, healthcare, industrial monitoring, personal health [17].

LoRa is complementing traditional cellular and short-range wireless protocols by addressing their shortcomings in an open and reliable matter with the detriment of low data rates as shown in the Figure 1.

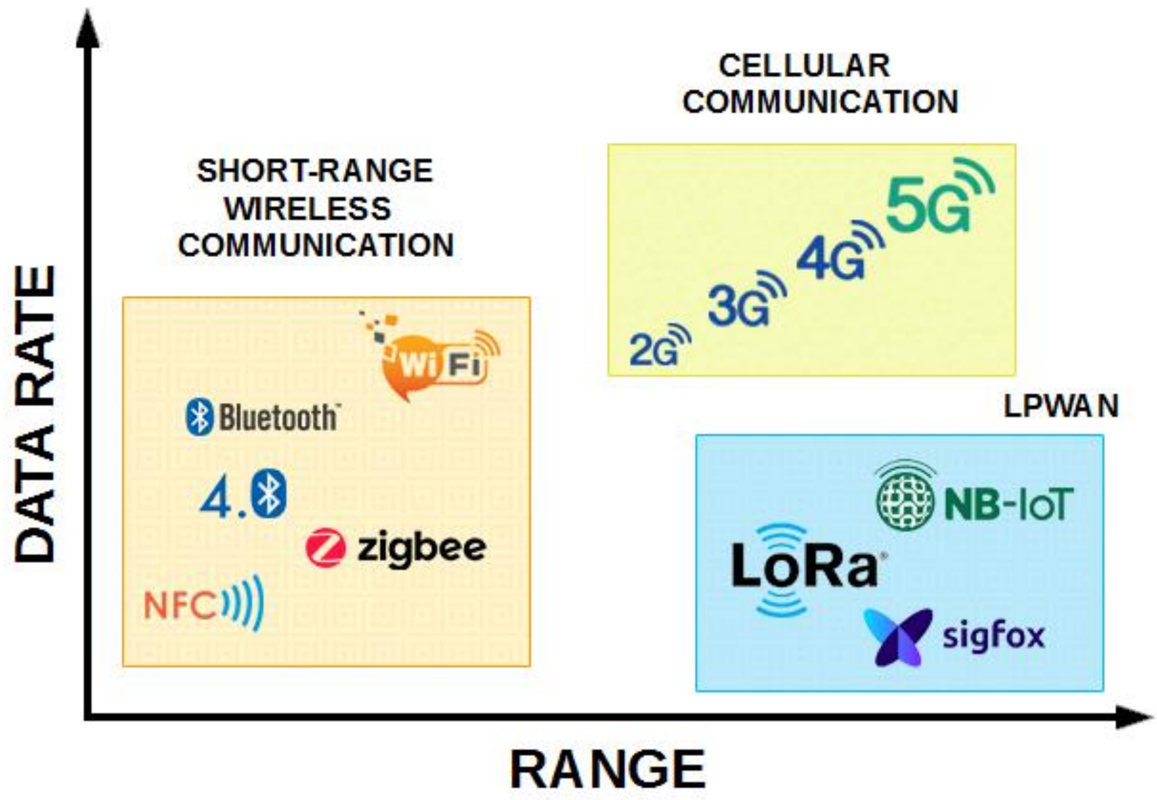


Figure 2 - Communication Protocols Range over Data Rate Plot [18]

LPWAN is unique because it takes different trade-offs than traditional technologies such as Personal Area Network (Bluetooth), ZigBee, Traditional Cellular, Local Area Network (Wi-Fi), Narrow-Band IoT and others, as see in Figure 2. [19]

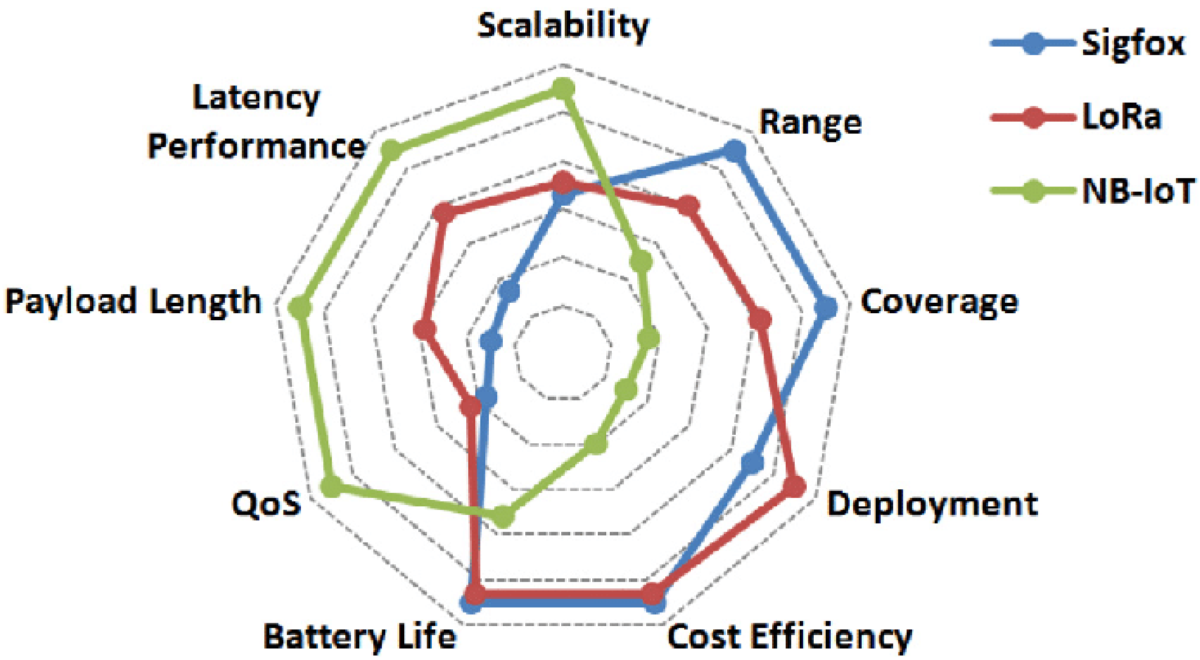


Figure 3 - Long Range Protocols Capabilities and Trade-offs [20]

Thus, LoRa offers wide-area wireless connectivity for low power devices, which is not provided by most other legacy technologies. The number of connected devices that are connected to the internet using long range communication is estimated to a quarter of the 30 thousand million IoT devices. [21]

LoRa real-world applicability

It would be impossible to talk about the relevance of LoRa Key Generation without the relevance of LoRa. Therefore, below are a few real-life reasons why LoRa has been selected as the IoT wireless communication protocol in this project.

In areas with geographical or environmental barriers, such as mountain ranges, heavy forests or just areas that have underdeveloped infrastructures, it is more suitable to rely on a technology such as LoRa. Some but not all the reasons for choosing LoRa in such a scenario are the following:

1. It is cheaper, one of the reasons underdeveloped areas are struggling with infrastructure is because of the unaffordable price. This is not the case for LoRa since it can be implemented and operated with limited resources.
2. As discussed, since infrastructure is limited, it means that interferences are also few. This means that a longer-range connection can be established. Depending on the region, a wireless connection that is up to 15km long can be established, and in some cases this value might be greater.
3. This technology is open source and is not regulated by a 3rd party contractor that might decide to suspend their services. This means that this critical infrastructure will not be suspended even if the parent company becomes bankrupt or is just unable to provide its services to the user for whatever reason.

Societal Impact

To explain how Wireless LoRa Key Generation can be useful, a scenario will be detailed below to create a structured understanding of why LoRa is the best option, and why Key Generation is required to ensure a secure and reliable connection.

In a country known to the general public as dangerous, such as Afghanistan, few well established companies would want to collaborate to develop the infrastructure for IoT devices.

- Therefore, highly regulated long-range protocols such as Traditional Cellular, Cat-M1 and Sigfox are out of question, since these companies would require operating in the respective country and the danger for the staff and reputation of the company might be too great.
- Much of the infrastructure is lacking and is one of the main problems and dominant reasons for its collapse. LoRa communication could be a way to ensure communication between the many remote regions of the country. This could at least solve the problem of communication for the vast mountainous range that the county is situated on.
- Security is an issue since, eavesdroppers are everywhere in a high conflict country where everyone wants to use every advantage they have. This is where messages that are sent insecurely can have grave consequences. Starting from simply stealing information, to tempering with the messages and interfering with communication altogether.

It is especially dangerous if false messages can be replaced and sent instead of the original ones. This can happen if the eavesdropper knows the key or there is no key being used at all. LoRa itself is simply an open-source protocol with no security implemented. Key Generation comes in to solve this issue.

Why is Key Generation important?

The emphasis on this project is the security aspect of the wireless communication. Assuming that the world is completely made of compassionate altruists, no security would ever be needed, and this project would be inexistent or simply obsolete.

The used keys are kept secretly within the devices once they are generated, they can be used to ensure the security of the message until a security breach (which usually happen because of human mistakes) happens and the keys need to be updated or regenerated. This means that eavesdroppers will struggle to figure out what the data is supposed to represent.

Therefore, it is essential to specify that security is a tremendous requirement when talking about implementing this technology. Key Generation assures that no information leakage is possible and with an insignificant power and time cost, keys can be generated to encrypt or decrypt plaintexts, or to verify the integrity of the transmitted data using the hash key.

When talking about cypher keys, true randomness or entropy is a good thing because it is hard to predict or to crack through brute force attacks if the key is long enough. By default settings in the project's code, the keys take approximately three minutes to generate. If more randomness for the keys is required, a bigger time gap between the samples can be added to provide more random results assuming a more dynamic or chaotic environment. This means that it takes approximately three minutes to have a secure IoT connection. Time can be critical in critical infrastructure, and ensuring security in such a short time can be extremely important in some scenarios.

As mentioned, the algorithm is generating the matching keys in a short amount of time. This also means that the power consumption of the paired IoT devices is limited. Assuming that one device uses a USB port, as in this project's case, that is 5V with a maximum current draw of 500mA. So that is 2.5 Watts at best, or 5W for the two devices. So, to set up the wireless security for two IoT devices, the maximum power consumption is 5W times the time the devices run for, so three minutes divided by one hour or 60 minutes. This equals to a total power consumption of 250mW to ensure security.

Altogether one of the possible and useful practical industrial implementations for Wireless LoRa Key Generation is to establish a secure, reliable, cost, time, and power efficient communication technology for IoT devices in remote or isolated, low interference locations.

Theory

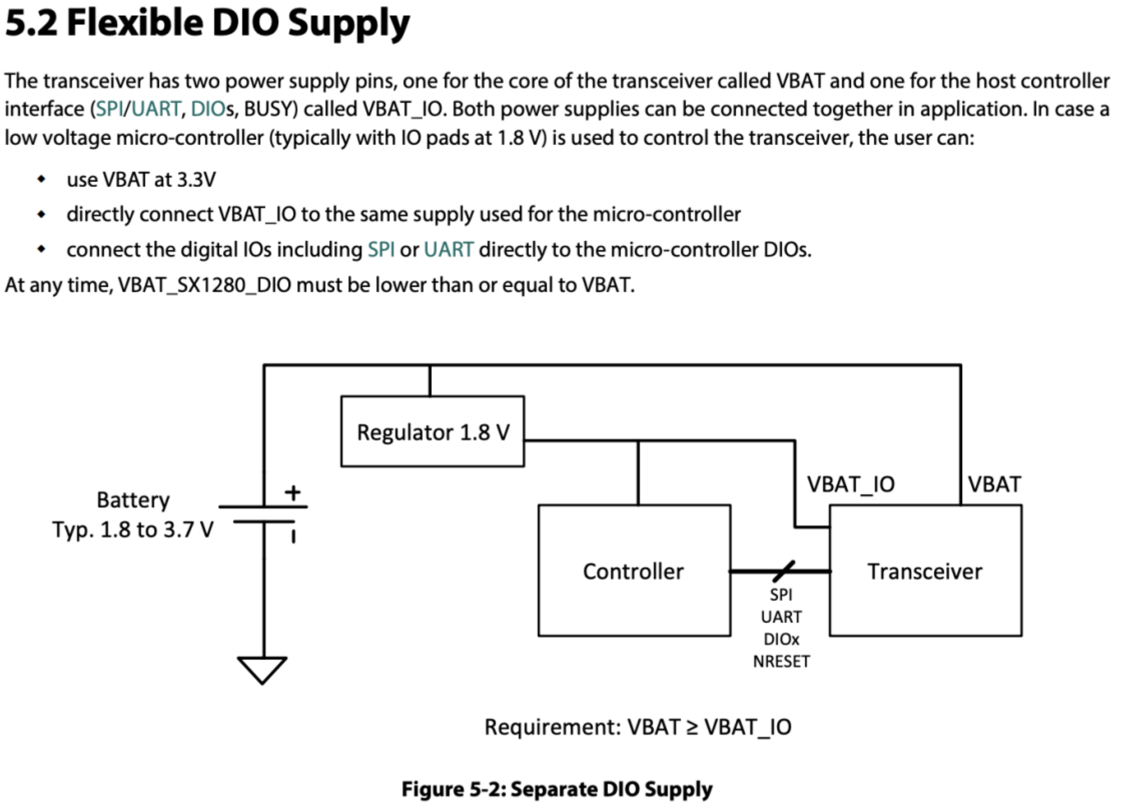
For the first weeks of the project, an in depth understanding of the project was required in order to start practical work. The time taken to understand Key Generation was slower than expected and from two weeks as planned for learning, it took as long as two months of research while concurrently working on the project.

Practical Results versus Theoretical Values for Hardware

First research about the LoRa protocol has been carried out. From this research a few theoretical outcomes have resulted. Some of these theoretical results and practical outputs are meant for the hardware capabilities such as the following:

The boards are expected to reach ranges between 3-5km in outdoor places with high electromagnetic interference. For concrete and steel buildings on multiple levels, this range can be gravely decreased. Most tests have been carried out inside the two libraries of the University of Liverpool and within the university's campus and in the proximity of a private accommodation. All attempts produced good results for all the test cases, however more packets were lost for when the distance between devices was increased, thus it took longer to generate the key, because packets needed to be resent. Therefore, the theoretical indoor distance value can be checked as valid. Tests for outside environments and distances as large as 3km could not be verified as it would have been risky to carry around a laptop on the street.

The power minimum power draw for the devices in theory is between 1.8V to 3.7V, as can be seen in the datasheet for the SX1280 in Figure 3.



5.2 Flexible DIO Supply

The transceiver has two power supply pins, one for the core of the transceiver called VBAT and one for the host controller interface (SPI/UART, DIOs, BUSY) called VBAT_IO. Both power supplies can be connected together in application. In case a low voltage micro-controller (typically with IO pads at 1.8 V) is used to control the transceiver, the user can:

- use VBAT at 3.3V
- directly connect VBAT_IO to the same supply used for the micro-controller
- connect the digital IOs including SPI or UART directly to the micro-controller DIOs.

At any time, VBAT_SX1280_DIO must be lower than or equal to VBAT.

Figure 5-2: Separate DIO Supply

Figure 4 - SX1280 Power Draw Datasheet [22]

The SX1280 board is a shield for the NUCLEO-L073RZ microcontroller. The name of this exact microcontroller is required for the configuration settings of the online programming IDE used. The software is written in Keil Studio. Figures 4 and 5 are images containing the pin layouts of the NUCLEO microcontroller.

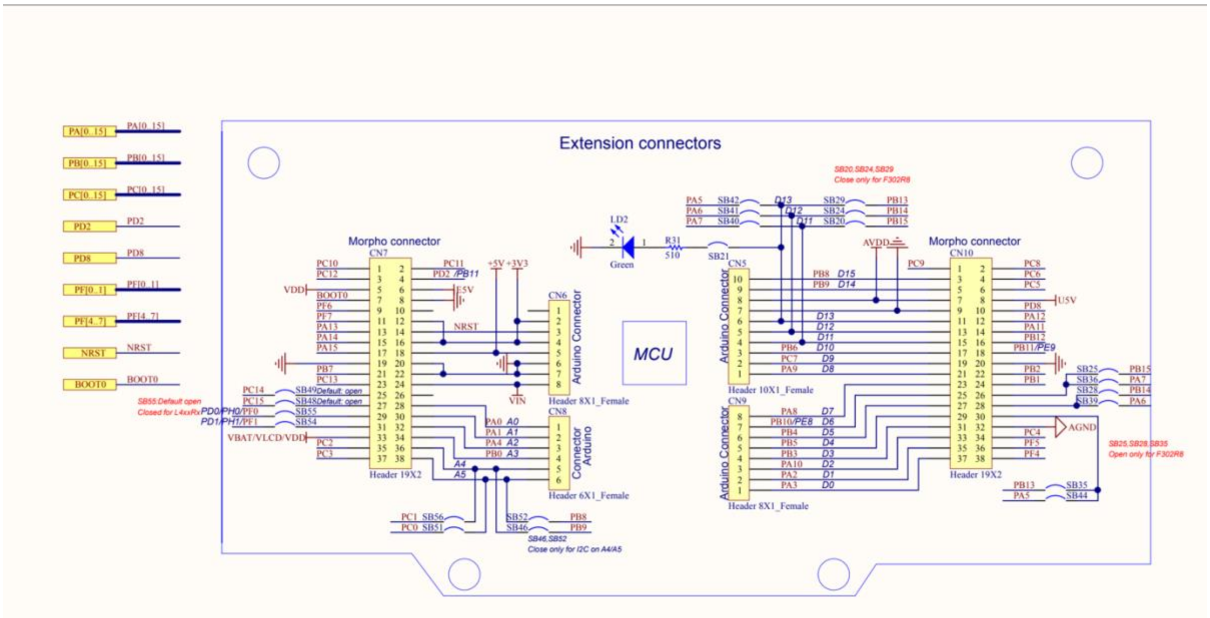


Figure 5 - NUCLEO-L073RZ Detailed Pin Layout

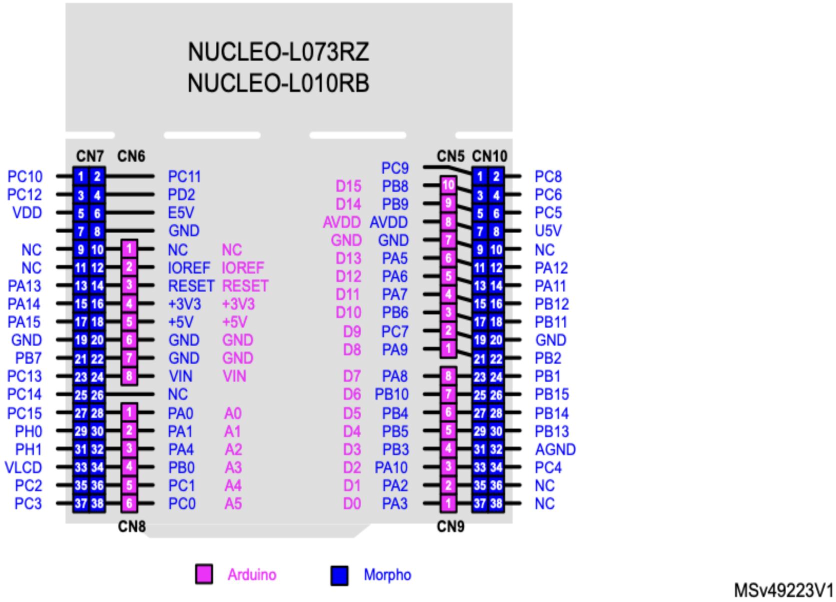


Figure 6 - NUCLEO-L073RZ General Pin Layout

Following the indication of the datasheets a USB port has been used to connect the devices to a computer. For the utilized MacBook, a USB to USB-C adapter has been acquired to make the connection possible. This allowed at most 5V with 500mA to be supplied.

For the algorithm power efficiency, the devices operate in sleep mode as often as possible. For example, during Channel Probing, once a device sends a message, it then goes into sleep mode until it receives a new message. Another example is once the Master produces the generated key and sends it to the Slave for Information Reconciliation, the Master then goes into sleep mode, and once the Slave finishes his entire Key Generation with the hash key generated as well, it also goes into sleep mode. A third and final example is that the Slave is always in sleep mode if it does not receive a message and it never sends a message without it receiving a message first. Therefore, both the hardware and the algorithms are optimised to run as power efficient as possible. Most debugging, especially in the early phases of the project's development has been done using the IO of the buttons and LEDs as presented in Figure 6.

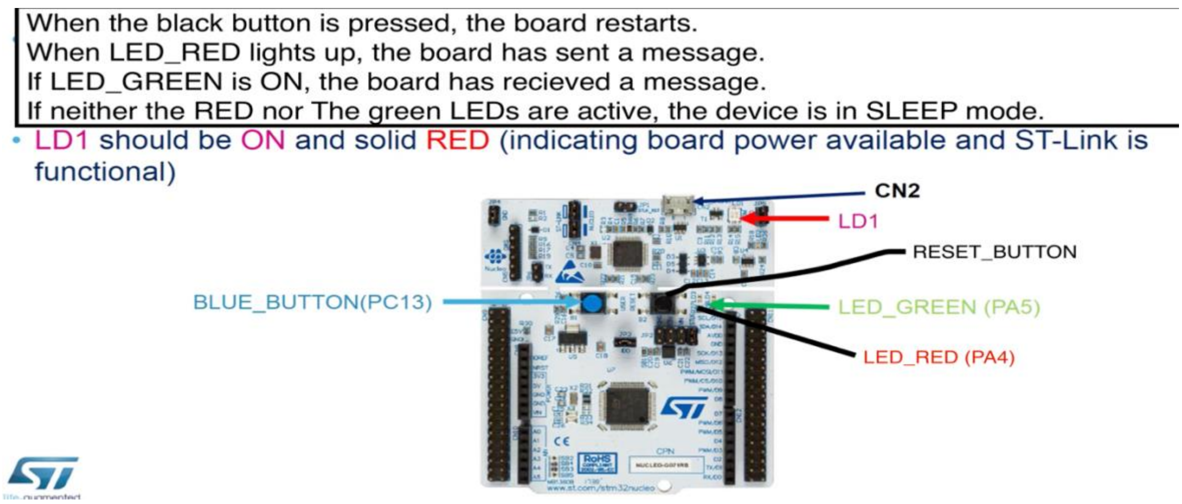


Figure 7 - Used Inputs and Outputs for NUCLEO-L073RZ [23]

Software Theoretical Objectives and Actual Implementations

As specified above, an MBED-compatible online environment is needed to develop the LoRa boards. The code of this project is written in C++ with the help of the Keil Studio online cloud environment.

The first step for debugging and understanding how the boards work was to download on the boards the publicly available, Ping-Pong algorithm from MBED [24]. The current algorithm of the project is a distant adaptation of the Ping-Pong implementation, but this first step was indeed the most important in the development process of the project.

To ensure the security of the cryptographic key, the stages for the key generation algorithm are needed on the software side of the project.

Below are the theoretical objectives and their results for the key generation algorithm:

A. The initial stages of the Channel Probing algorithm were arguably the easiest to implement. Because as explained above, to record several RSSI values, the Ping-Pong algorithm would have been sufficient since messages could be sent from one device to another and each of these devices would record the SNR or RSSI. This proved to be inadequate for a key generation algorithm that expects to produce keys of the same lengths and with the same binary values. To be certain that the keys are of the same size, there would be a need for one device to acknowledge that the other device has the same data packet. In this case, this algorithm is called Packet Matching and is used for validating that both IoT devices have received the same number of messages that will later be used to generate keys of the same size and similar enough for the information reconciliation algorithm.

B. Hence, for the Channel Probing algorithm, an innovative approach was required for validating the number of received packets for the two devices. In the main theoretical concept of the algorithm, an acknowledgement was needed to check the validity of each received packet. This confirmation is however currently unavailable for the LoRa side of the SX1280 library. Therefore, a new approach that does not require acknowledgements is needed. Counting the number of packets for each device and discarding the unwanted packets could be a solution to this problem. However, the available LoRa SX1280 library does not currently have a function for counting the number of received packets. Thus, the project has been developed and implemented a unique solution following the advice of the supervisor. To understand this new implementation, read the relevant paragraphs about the Packet Matching algorithm in the Design section of the report.

C. For Quantization, various algorithms and implementations could have been used, such as Scalar, Adaptive, Robust, or MMSE (Minimum Mean-Square Error) quantization. Differential Based Quantization has been used because it is one of the most simple, reliable, and easy-to-implement uniform quantization algorithms available, and it works perfectly with the requirements of this project. The idea of using a library for the quantization algorithm was tempting, however seeing that the work requirements are too specific, a personal C++ implementation of the below pseudocode algorithm for the mean-value based has been implemented.

INPUT: X_u %RSSI of user u

INPUT: ϵ %RSSI resolution

OUTPUT: K_u %Generated key sequence of user u

for $i \leftarrow 1$ **t**o $N-1$ **do**

if $X_u(i+1) > X_u(i) + \epsilon$ **then**

$K_u(i) = 1$

else if $X_u(i+1) < X_u(i) - \epsilon$ **then**

$K_u(i) = 0$

else

$X_u(i)$ dropped

end if

end for

Figure 8 - Differential Based Quantization Algorithm from Research Paper [25.1]

D. In theory, the information reconciliation algorithm should have looks like the pseudocode in the figure below. For this algorithm in theory Alice should send a syndrome to the Slave. This syndrome is the XORed key of the Alice with a random code from the BCH code pool. Once the syndrome is received by the Bob, he will find the decryption code from his own BCH code pool. Bob does this finding the message with the minimum Hamming Distance. Once Bob finds the correct code, he will XOR this code with the syndrome and find Alice’s key. Now both Alice and Bob have the same keys and the information is reconciliated. However just implementing the BCH code or using a library would have provided nothing innovative to this project. This new algorithm is inspired by the procedures of the pseudocode below. Consequently, in the Design section of the report, the self-developed algorithm for Information

Reconciliation is presented in a clear flowchart.

INPUT: K_A, K_B %Quantized keys of Alice and Bob

INPUT: C %ECC set shared by Alice and Bob

OUTPUT: K_A, $K_{B'}$ %Reconciled key

1: Alice randomly selects a code c from an ECC set C

2: Alice calculates $s = \text{XOR}(K_A, c)$

3: Alice transmits s to Bob through a public channel

4: Bob receives s

5: Bob calculates $c_B = \text{XOR}(K_B, s)$

6: Bob decodes c_B to get c%When $dis(c - c_B) < t$

7: Bob calculates $K_{B'} = \text{XOR}(c, s) = K_A$ %Alice and Bob agree on the same key

Figure 9 - Information Reconciliation Pseudocode from Research Paper [25.2]

E. The final theoretical practice for the key generation algorithm is Privacy Amplification. To understand the working principle of Privacy Amplification, research about Hash Keys and Hashing algorithms has been carried out. Hash keys exist to provide a unique secret hashed key for checking that the devices hold the same key without the risk of displaying and compromising the keys. For this step, a new algorithm was not required since the pre-existing algorithms are suitable for the requirements of this project. Many C++ libraries could have been used, the one included in this algorithm is Mbed-TLS [25]. Out of this library, SHA-256, or the Bitcoin's Algorithm [26] is used, as it provides more security than SHA-128 or MD-5. It is said that SHA-256 has not been compromised as of yet even with the currently available supercomputers [27].

Design

Packet Matching Algorithm Design Implementations and Results

As described in the Theory chapter, for Channel Probing, the initial plan for Packet Matching was to make an acknowledgement-based validation algorithm.

Initially seeing that the SX1280 C++ library does not offer a function for Acknowledgments or for counting the number of received packets, with the advice of the Supervisor, it was decided to try and modify a byte in the header to make it act as an acknowledgement. This was however a misunderstanding from my side and too difficult to accomplish, so it has been agreed to move to a different approach.

In short, with this new approach, the slave would instead send his currently received packet number which will be checked at the master's end and be saved or not, based on validating the number. This process is explained in the Experimental method chapter of the report. The first sketches of this approach can be seen in the figure below.

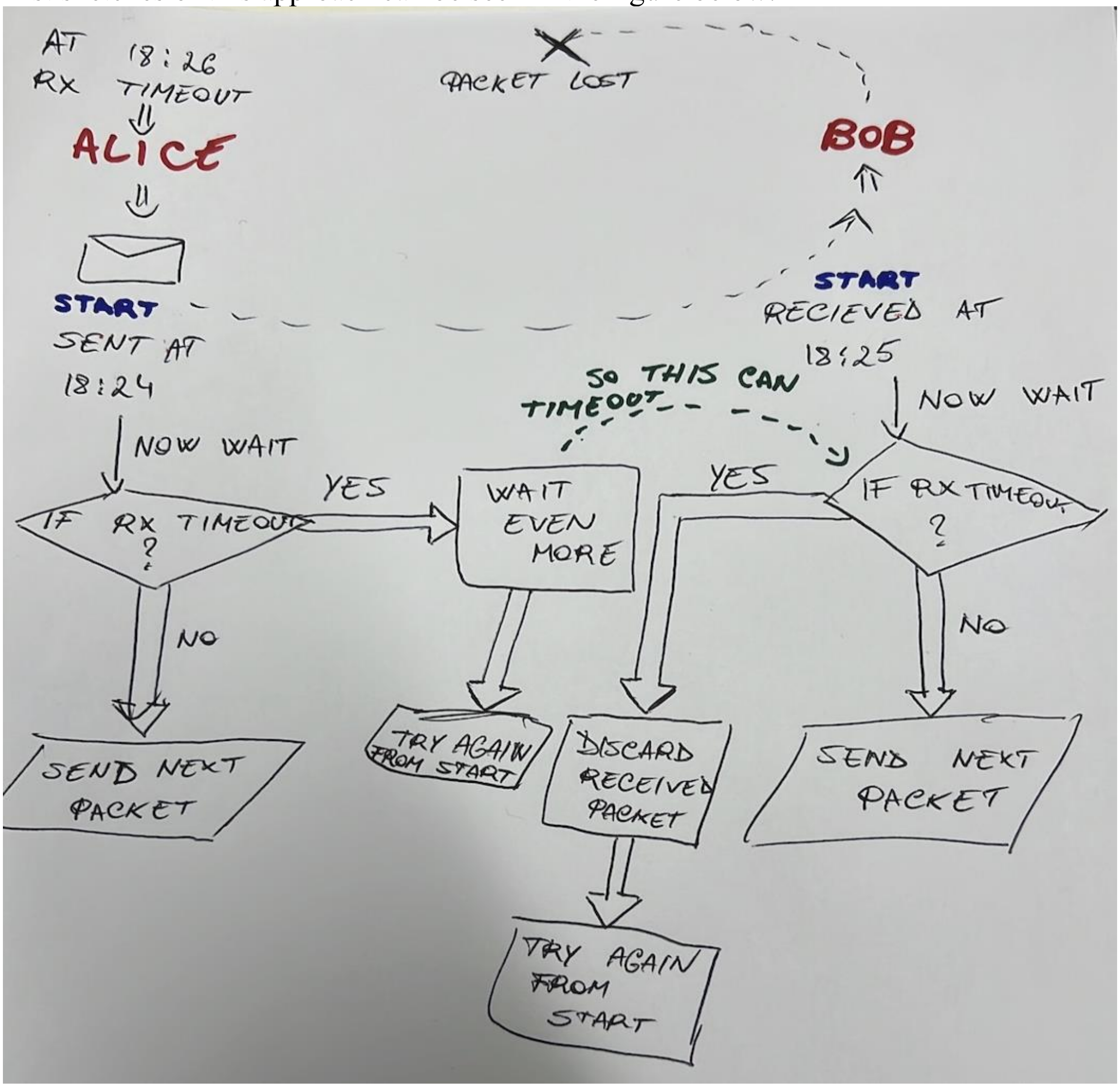


Figure 10 - Initial Sketch Flowchart for Packet Matching Algorithm

As suggested by this project's assessor, the best approach to describe the workings of an algorithm is through a flow chart. Provided below is the flow chart for the self-created packet matching algorithm in its final form.

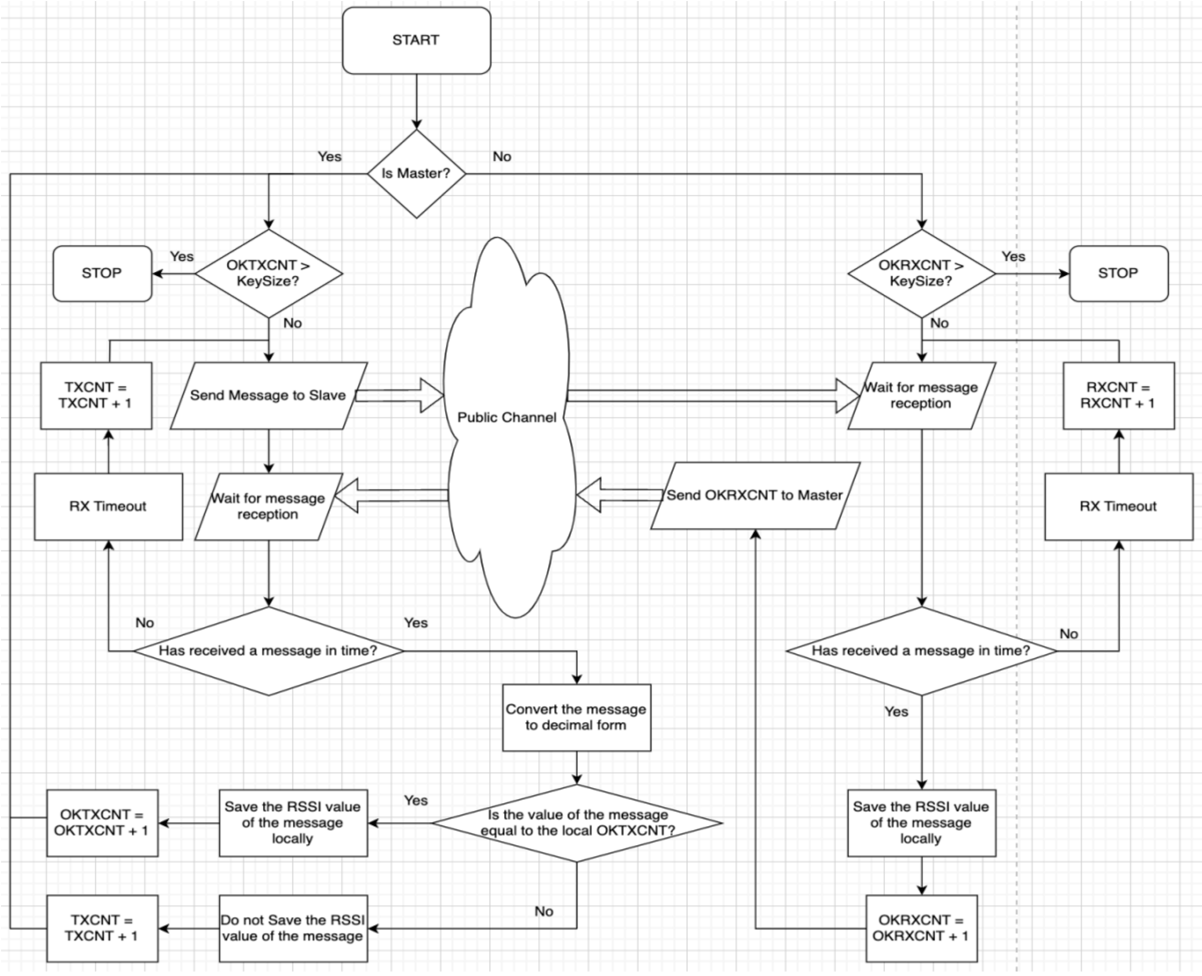


Figure 11 - Flowchart of the original design for the Packet Matching Algorithm

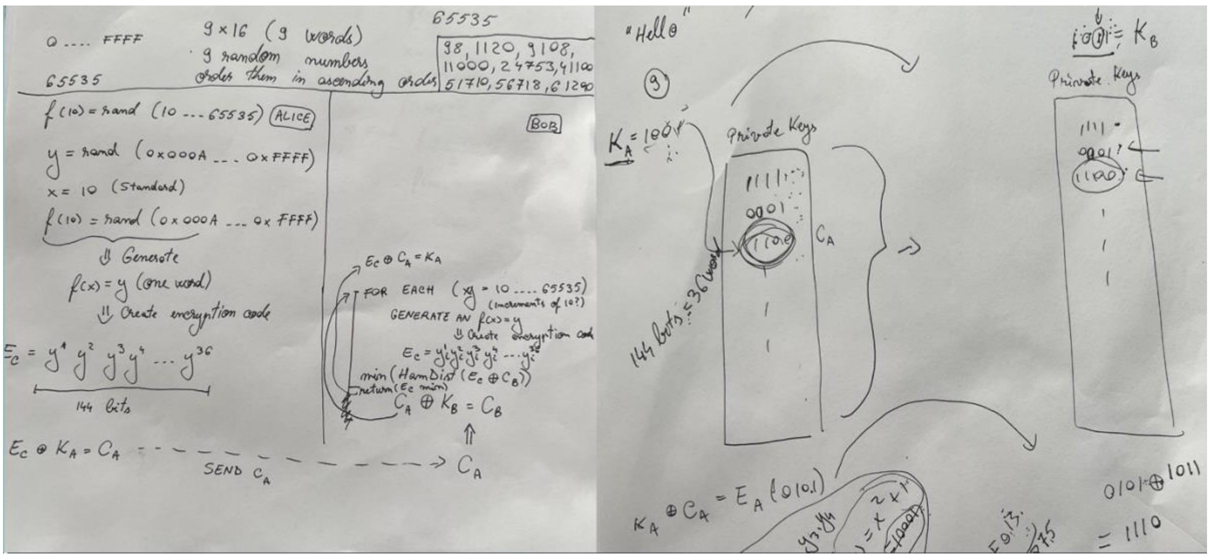


Figure 12 - Early Sketch Information Reconciliation Algorithm

Information Reconciliation Algorithm Formulas

Information Reconciliation formula for brute forcing the syndrome:

$$\boldsymbol{P} = \frac{\mathbf{1}}{\boldsymbol{c^m}} \times \mathbf{100\%}$$

Information Reconciliation formula for maximum percentage of errors:

$$\boldsymbol{MEp} = \frac{\mathbf{1}}{\boldsymbol{c}} \times \mathbf{100\%}$$

Where:

i. **P** = probability of cracking the code
ii. **MEp** = maximum error percentage
iii. **c** = predefined constant for the multifurcating tree size, $\forall c \in \mathbb{N}$
iv. **k** = key size, $\forall k \in \mathbb{N}$
v. **w** = word size, $\forall w \in \mathbb{N}$
vi. **m = k / w**, ratio between the key size ***k*** and word size ***w***

Information Reconciliation Algorithm Calculations for Design

Now the formulas will be put to the test for the current settings of the key generation algorithm.

Given any k = 144 bits long syndrome, with w = 16 bits words and a predefined constant c = 6, there are:

i. **m = 144 / 16 = 9 words**
ii. $\boldsymbol{MEp} = \frac{\mathbf{1}}{\mathbf{6}} \times \mathbf{100\%} \cong \mathbf{17\%}$
iii. $\boldsymbol{P} = \frac{\mathbf{1}}{\mathbf{6^9}} \times \mathbf{100\%} = \boldsymbol{0.00000992\%}$

Assuming that no information is leaked when sending this key, no more than once, the Information Reconciliation algorithm is secure.

Mathematical Induction for Information Reconciliation Probability Formula

Let the mathematical induction start with the initial case of a six-nary tree.

Starting with the least significant 16 bits of the 144-bit key code, for each selection the probability of finding the correct code decreases by a factor of pow 6.

So, there is logarithmic decrease of finding each correct word with the formula.

The probability of finding the right encryption code without knowing Bob's key is 1/6 for the least significant 16 bits.

With the use of the correct code, compute another 6 codes out of which only one is correct with the following word of the syndrome (or with the next 16 least significant bits).

The probability now of finding the correct least significant 32 bits without knowing Bob's key is $1/6 \times 1/6 = 1/6^2$.

Using the same principle, with the knowledge of the correct next 16-bit encryption code, compute another 10 encryption codes.

The probability now of finding the correct least significant 48 bits without knowing Bob's key is $1/6 \times 1/6 \times 1/6 = 1/6^3$.

As a general case, the probability of finding the correct number is:

1 in $1/6 \times 1/6 \times 1/6 \times \ldots \times 1/6 \times 1/6 \times 1/6 = 1/6^m$, where m is the number of words,

or $\boldsymbol{P} = \frac{\mathbf{1}}{\mathbf{6^m}} \times \mathbf{100\%}$ as a probability.

In this it was chosen that c = 6, but this holds $\forall c \in \mathbb{N} \Longrightarrow \boldsymbol{P} = \frac{\mathbf{1}}{\boldsymbol{c^m}} \times \mathbf{100\%}$

A visual explanation of another rootless binary-tree mathematical induction for the probability formula can be viewed in the figure below:

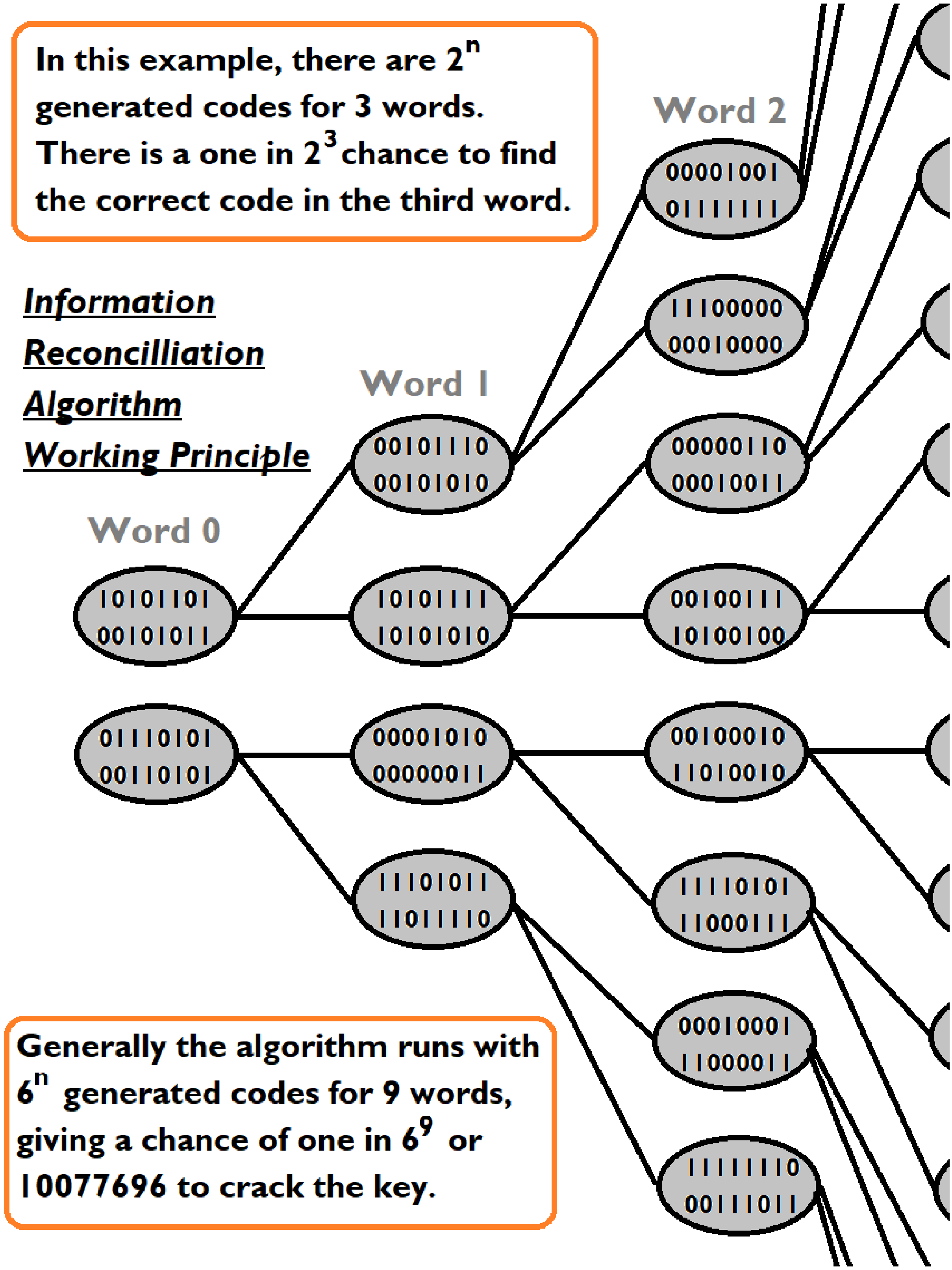


Figure 13 - Information Reconciliation Algorithm Complexity for Rootless Binary Tree Generation

Experimental method

Experimental Implementation and Explanation for the Packet Matching Algorithm

As presented in the Design chapter of the report and described in the flowchart, a new algorithm was used for Packet Matching. This new design as presented in the flowchart involved one node to send its currently counted packet number. The workings of this algorithm were mainly tested in the Harold Cohen and Sydney Jones libraries in the University of Liverpool campus. To identify as many errors as possible, often one IoT device was left in a room to record data on a computer's serial monitor, and the other device would be moved around over the floors of these libraries.

One of these early tests for the Packet Matching algorithm is explained below:

1. First one side of the Alice-Bob network needed to send their own received packet value to ensure that the other side has the same local received packet value, and in the case that there is a mismatch in the received packet number, to discard all unmatching packets, this can be clearly seen on the master's terminal in Figure 7.
2. Alice's variable that counts the number of sent messages is called TXCNT and the number that tracks the number of successfully sent packets, or confirmed packets, is called OKTXCNT. In Figure 7, the 73 RSSI values saved in local memory are discarded. The RSSI values are not truly 'discarded' but are never saved. For clarity of explanation, the concept of discarding the packets is used.
3. For the Bob side of the wireless communication channel, instead of sending a message to Alice such as "Pong", Bob sends his current received packet number, such as "840". This value is named OKRXCNT and can be seen in practice in Figure 8.
4. Bob never sends a message to Alice unless he has received a message from Alice first. So, Bob works in the manner "Talk only if you are asked". Bob will never send a message unless he received a message first. Thus, for N received messages, he sends back N messages.
5. Once Alice receives Bob's message (which is Bob's OKRXCNT), she compares this value to her local TXCNT variable, which is visible in Figure 7. This way Alice can see if she has the correct packets in her local vector. If she has "73" in her local TXCNT variable, and the received message is "1", then she discards the 72 unmatching packets.

Altogether this methodology has been tested under various circumstances, is practical and usable in all scenarios. In theory everything is logically correct and in practice it works as expected. This design is mathematically elegant, compact, and original.

Figures 8 and 9 should be interpreted as a pair since it is the same test for the Alice and Bob terminals.

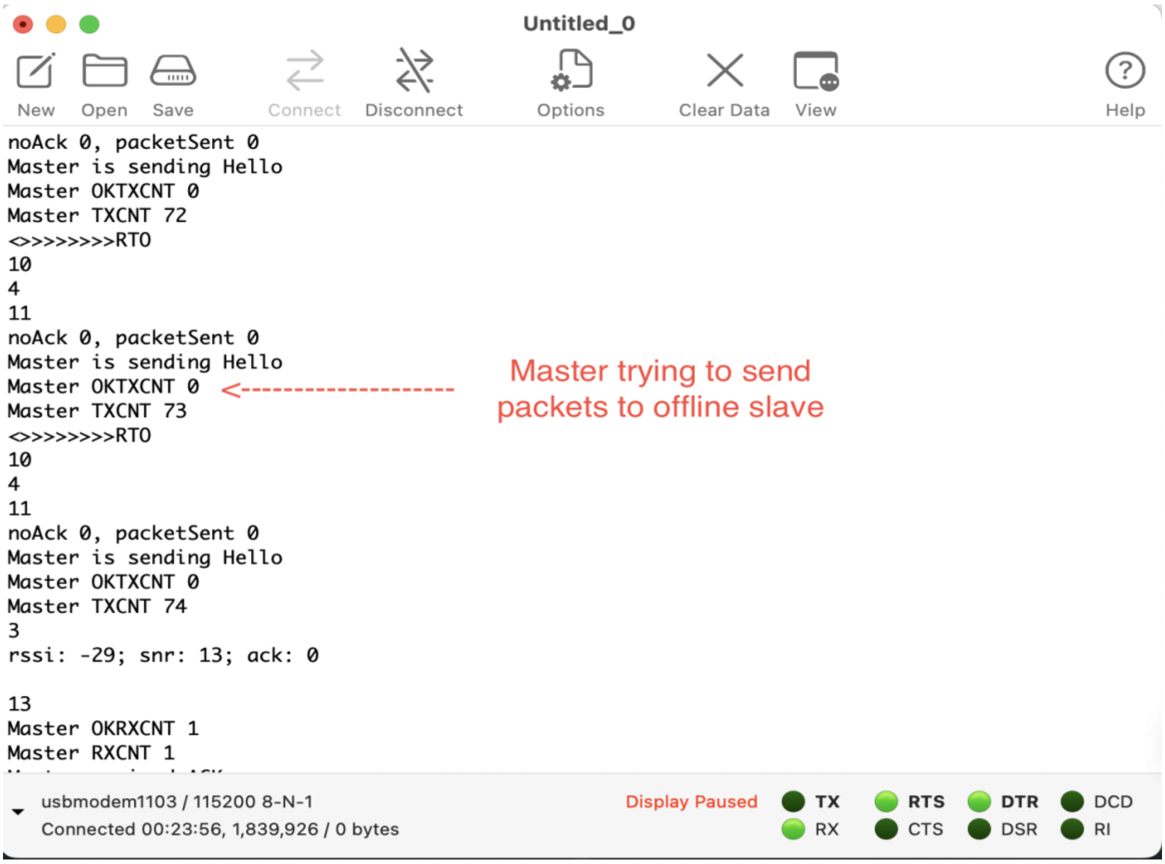


Figure 14 - Master Terminal Discarding Packets Sent to Offline Slave

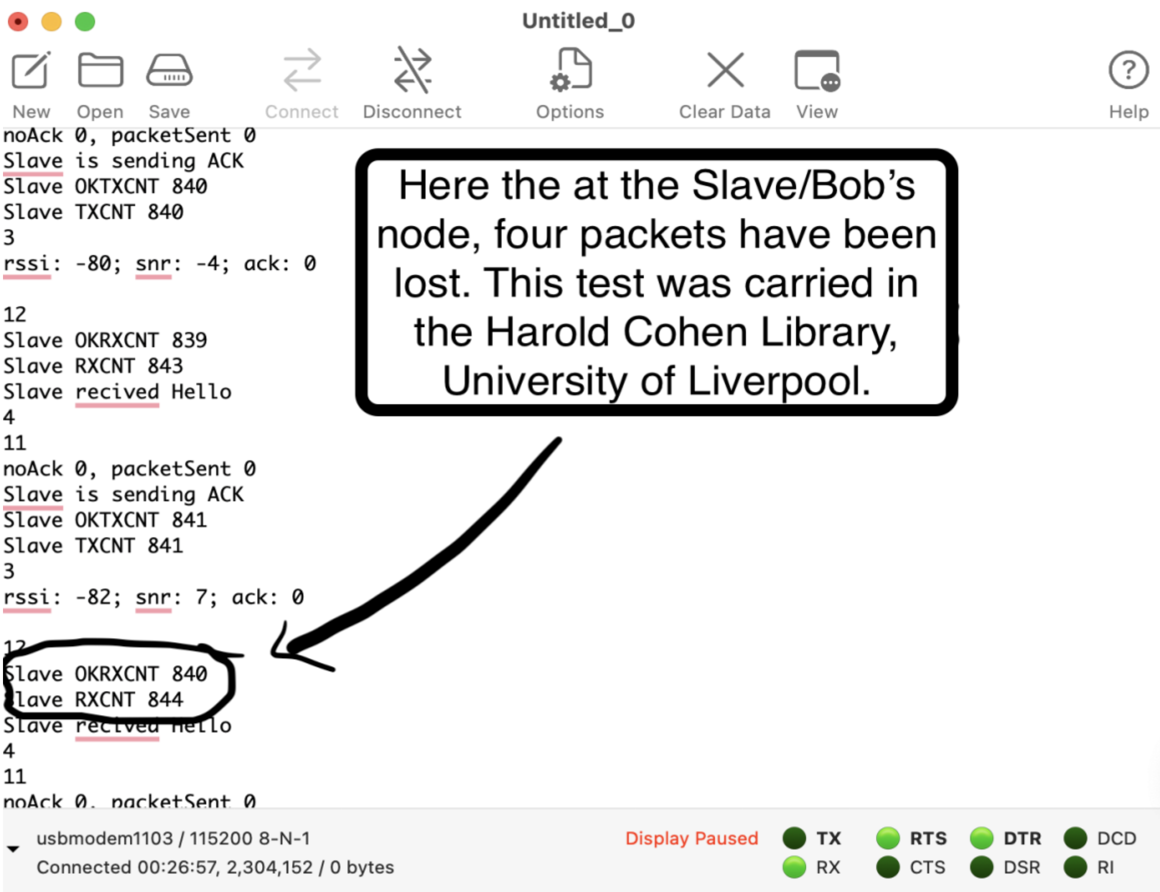


Figure 15 - Slave Terminal Lost Packets for Active/Online Alice-Bob Communication

This way the received packets for both Alice and Bob are the same and the packet matching algorithm is ensured to work even though the SX1280 library does not provide any functionalities for the working of this key generation requirement.

In the ideal case presented in the image below, no packets are recorded as lost on Bob's serial monitor.

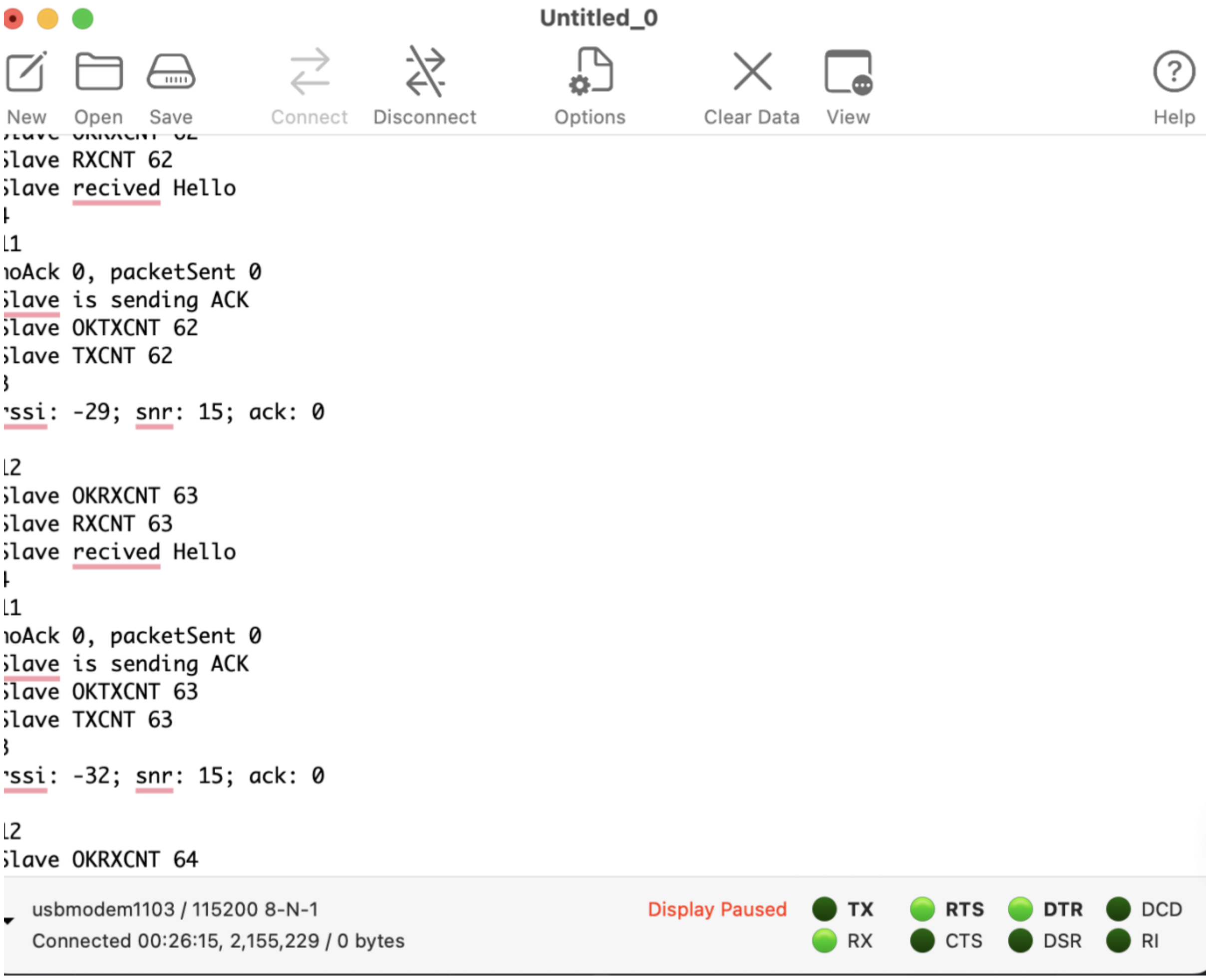


Figure 16 - Ideal case of no lost packets at Bob's terminal

Below is the pool of random numbers. They are generated by only choosing numbers whose digits sum up to ten. So, for 1054, 1 + 0 + 5 + 4 = 10. But any method of generating big and varied numbers can be chosen so that the binary versions of the decimal numbers do not have too many coinciding bytes. The list used for this algorithm contains 963 numbers where 0xFFFF is the upper limit. In this case the upper limit is 64000, because this is the biggest decimal number whose digits sum up to 10. In total there are 963 numbers that hold this property between 0x0000 to 0xFFFF. This method has been chosen also because the hardware devices do not allow more than 1024 elements in a single vector and this algorithm for generating the pool is varied and big enough for a vector of maximum 1024 elements.

```
19 28 37 46 55 64 73 82 91 109 118 127 136 145 154 163 172 181 190 208 217 226 235 244 253 262 271 280 307 31
6 325 334 343 352 361 370 406 415 424 433 442 451 460 505 514 523 532 541 550 604 613 622 631 640 703 712 721
730 802 811 820 901 910 1009 1018 1027 1036 1045 1054 1063 1072 1081 1090 1108 1117 1126 1135 1144 1153 1162
1171 1180 1207 1216 1225 1234 1243 1252 1261 1270 1306 1315 1324 1333 1342 1351 1360 1405 1414 1423 1432 1441
1450 1504 1513 1522 1531 1540 1603 1612 1621 1630 1702 1711 1720 1801 1810 1900 2008 2017 2026 2035 2044 2053
2062 2071 2080 2107 2116 2125 2134 2143 2152 2161 2170 2206 2215 2224 2233 2242 2251 2260 2305 2314 2323 2332
2341 2350 2404 2413 2422 2431 2440 2503 2512 2521 2530 2602 2611 2620 2701 2710 2800 3007 3016 3025 3034 3043
3052 3061 3070 3106 3115 3124 3133 3142 3151 3160 3205 3214 3223 3232 3241 3250 3304 3313 3322 3331 3340 3403
3412 3421 3430 3502 3511 3520 3601 3610 3700 4006 4015 4024 4033 4042 4051 4060 4105 4114 4123 4132 4141 4150
4204 4213 4222 4231 4240 4303 4312 4321 4330 4402 4411 4420 4501 4510 4600 5005 5014 5023 5032 5041 5050 5104
5113 5122 5131 5140 5203 5212 5221 5230 5302 5311 5320 5401 5410 5500 6004 6013 6022 6031 6040 6103 6112 6121
6130 6202 6211 6220 6301 6310 6400 7003 7012 7021 7030 7102 7111 7120 7201 7210 7300 8002 8011 8020 8101 8110
8200 9001 9010 9100 10009 10018 10027 10036 10045 10054 10063 10072 10081 10090 10108 10117 10126 10135 10144
10153 10162 10171 10180 10207 10216 10225 10234 10243 10252 10261 10270 10306 10315 10324 10333 10342 10351 1
0360 10405 10414 10423 10432 10441 10450 10504 10513 10522 10531 10540 10603 10612 10621 10630 10702 10711 10
720 10801 10810 10900 11008 11017 11026 11035 11044 11053 11062 11071 11080 11107 11116 11125 11134 11143 111
52 11161 11170 11206 11215 11224 11233 11242 11251 11260 11305 11314 11323 11332 11341 11350 11404 11413 1142
2 11431 11440 11503 11512 11521 11530 11602 11611 11620 11701 11710 11800 12007 12016 12025 12034 12043 12052
12061 12070 12106 12115 12124 12133 12142 12151 12160 12205 12214 12223 12232 12241 12250 12304 12313 12322 1
2331 12340 12403 12412 12421 12430 12502 12511 12520 12601 12610 12700 13006 13015 13024 13033 13042 13051 13
060 13105 13114 13123 13132 13141 13150 13204 13213 13222 13231 13240 13303 13312 13321 13330 13402 13411 134
20 13501 13510 13600 14005 14014 14023 14032 14041 14050 14104 14113 14122 14131 14140 14203 14212 14221 1423
0 14302 14311 14320 14401 14410 14500 15004 15013 15022 15031 15040 15103 15112 15121 15130 15202 15211 15220
15301 15310 15400 16003 16012 16021 16030 16102 16111 16120 16201 16210 16300 17002 17011 17020 17101 17110 1
7200 18001 18010 18100 19000 20008 20017 20026 20035 20044 20053 20062 20071 20080 20107 20116 20125 20134 20
143 20152 20161 20170 20206 20215 20224 20233 20242 20251 20260 20305 20314 20323 20332 20341 20350 20404 204
13 20422 20431 20440 20503 20512 20521 20530 20602 20611 20620 20701 20710 20800 21007 21016 21025 21034 2104
3 21052 21061 21070 21106 21115 21124 21133 21142 21151 21160 21205 21214 21223 21232 21241 21250 21304 21313
21322 21331 21340 21403 21412 21421 21430 21502 21511 21520 21601 21610 21700 22006 22015 22024 22033 22042 2
2051 22060 22105 22114 22123 22132 22141 22150 22204 22213 22222 22231 22240 22303 22312 22321 22330 22402 22
411 22420 22501 22510 22600 23005 23014 23023 23032 23041 23050 23104 23113 23122 23131 23140 23203 23212 232
```

Figure 17 - Pool of 963 Generated Numbers whose Digits Sum Up to 10

```
60121 60130 60202 60211 60220 60301 60310 60400 61003 61012 61021 6
2002 62011 62020 62101 62110 62200 63001 63010 63100 64000
Number 1 as binary  0101101111001100 index:631  as decimal:23500
Number 2 as binary  0010011110111011 index:299  as decimal:10171
Number 3 as binary  1110101001111111 index:930  as decimal:60031
Number 4 as binary  0101011011001110 index:598  as decimal:22222
Number 5 as binary  0001101101110110 index:266  as decimal:7030
Number 6 as binary  1100011110011111 index:897  as decimal:51103
Number 7 as binary  0101001011110000 index:565  as decimal:21232
Number 8 as binary  0001001111110000 index:233  as decimal:5104
Number 9 as binary  1010110001000101 index:864  as decimal:44101
Number 10 as binary  0100111110110100 index:532  as decimal:20404
Random code: 0011001011101001
```

Figure 18 - Randomly Chosen 10 Numbers from the 963 Number Pool

Results and Calculations

Testing the Channel Probing and Quantization Algorithms

The test results are displayed in this section of the report.

For the first test (Test 1), the IoT devices were distanced from one another periodically. In this test the two devices were probed to be distanced at equal time intervals so that half of the time they are farther from one another and half of the time in close proximity. The goal of this test is to check that the recorded RSSI values are accurate and that the Quantization algorithm records zeros half of the time and ones the other half. Something similar to a square wave, or in binary like …00001111000011110000111 1....

Below is the plot of this first test:

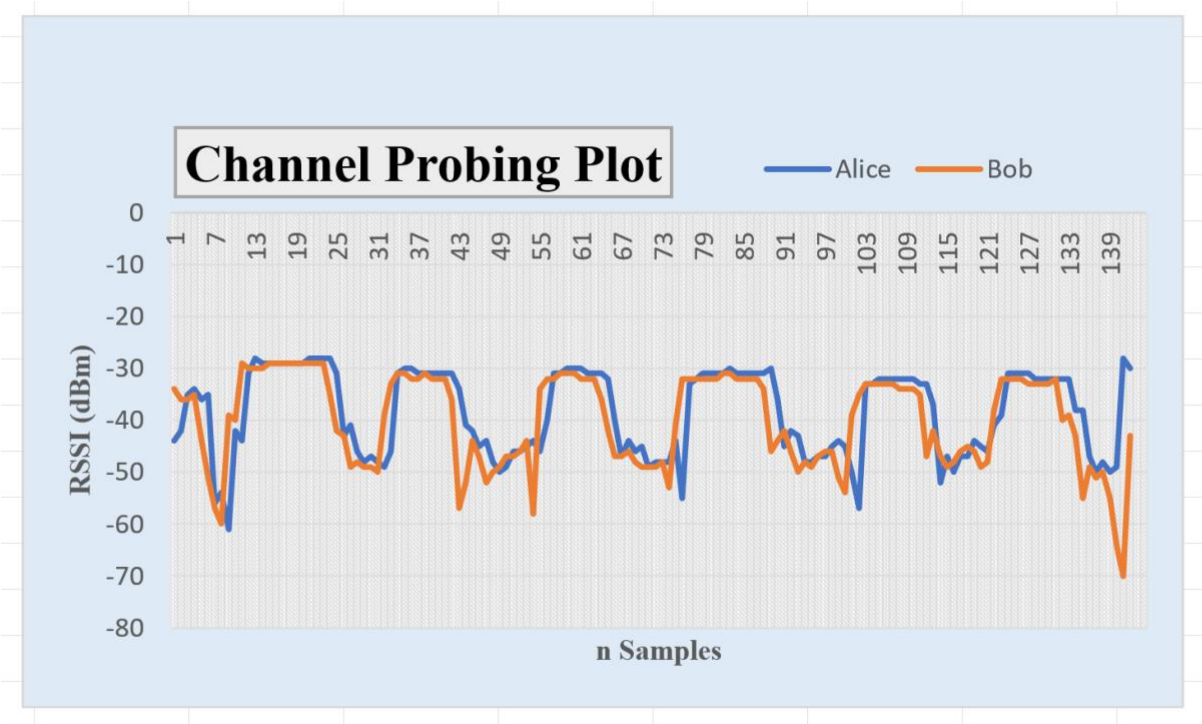


Figure 19 - Test 1 - Channel Probing Plot for Periodically Distancing the Devices

For this first test, in the below image is the dataset of RSSI values for the Alice and Bob serial monitors and their own generated quantized keys:

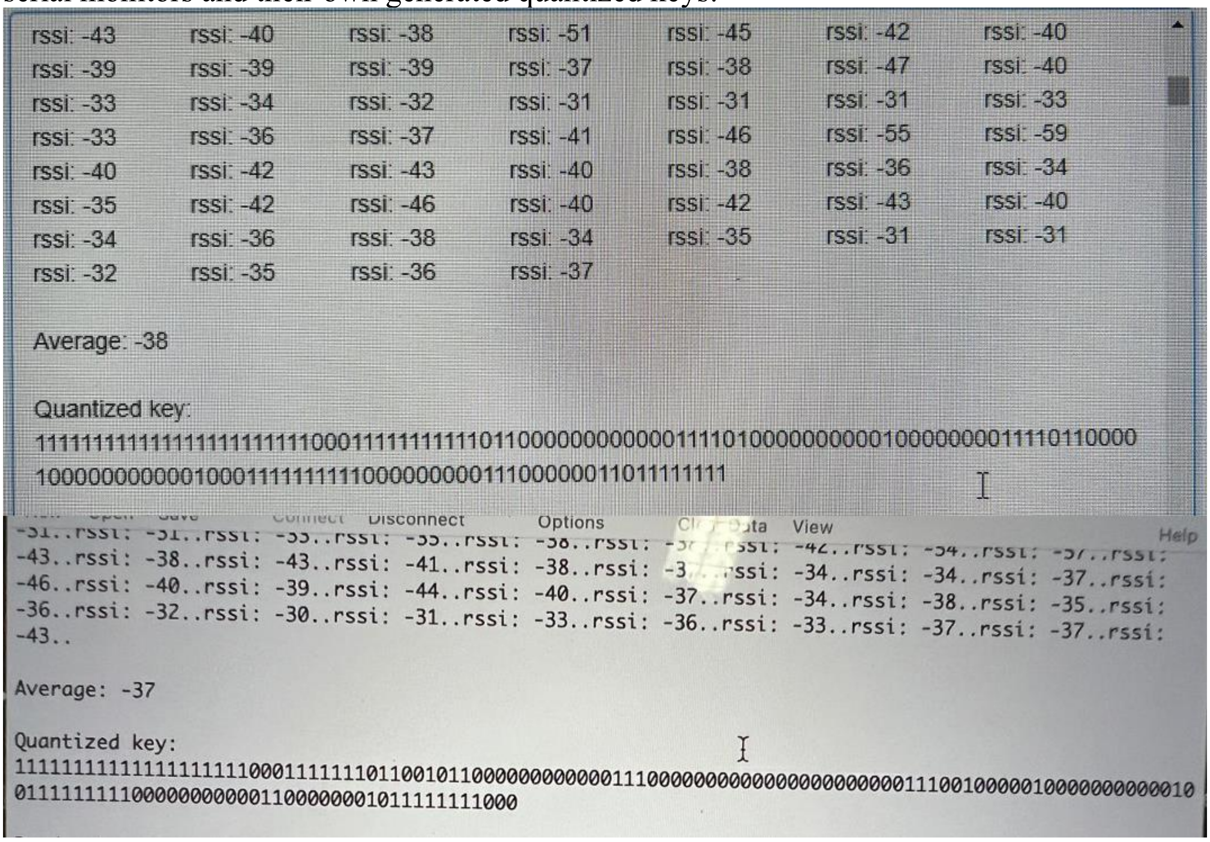


Figure 20 - Test 1 - Channel Probing and Quantization Serial Monitors Datasets for Periodically Distancing the Devices

Once it has been determined that the first two key generation algorithm steps are functional, further experiments have been completed to prove that the Packet Matching algorithm works in every case, and not just in some specific tests.

Testing the Packet Matching Algorithm

In the second experiment (Test 2), one device was left to record data on a stationary desktop, and the other device was carried around the 3rd and 4th floors of the Electrical and Electronics Engineering department of the University of Liverpool.
As seen in the below image even though the RSSI values go down to as low as -80dBm, there is still the same number of validated packets for both Alice and Bob, proving that the Packet Matching algorithm works in more varied and general test scenarios.

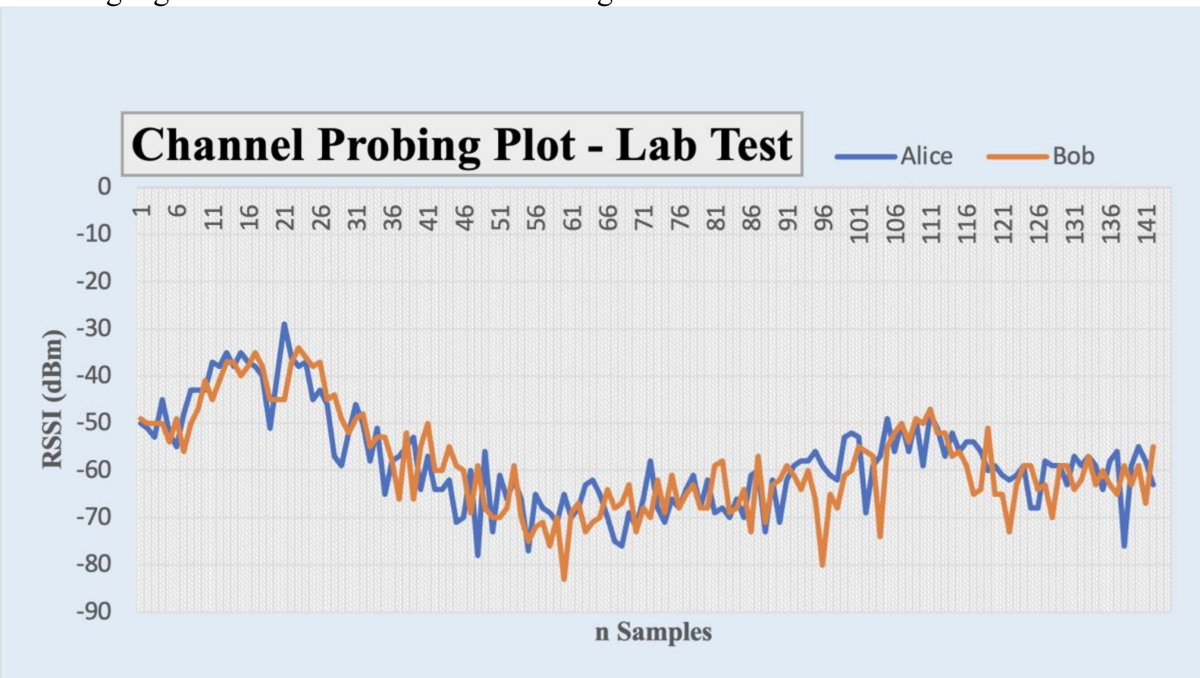


Figure 21 - Test 2 - Channel Probing Plot for Randomly Distancing the Devices while Walking in the Laboratory Floors

In the third test (Test 3), both devices started reception in the exact same instant. By having the master stationary on a desktop on the 2nd floor and the other IoT device moving in and out of the 2nd floor lift of the Harold Cohen library of the University of Liverpool campus. Many packets lose have been recorded by doing the lift experiment. As both master and slave started communicating at the same moment, they both count the same number of lost packets. All the packets lost are the ones which the slave fails to receive from the master. By doing this third experiment, it is further proven that the Packet Matching algorithm always works as expected. The images of the two serial monitors are displayed in the figure below.

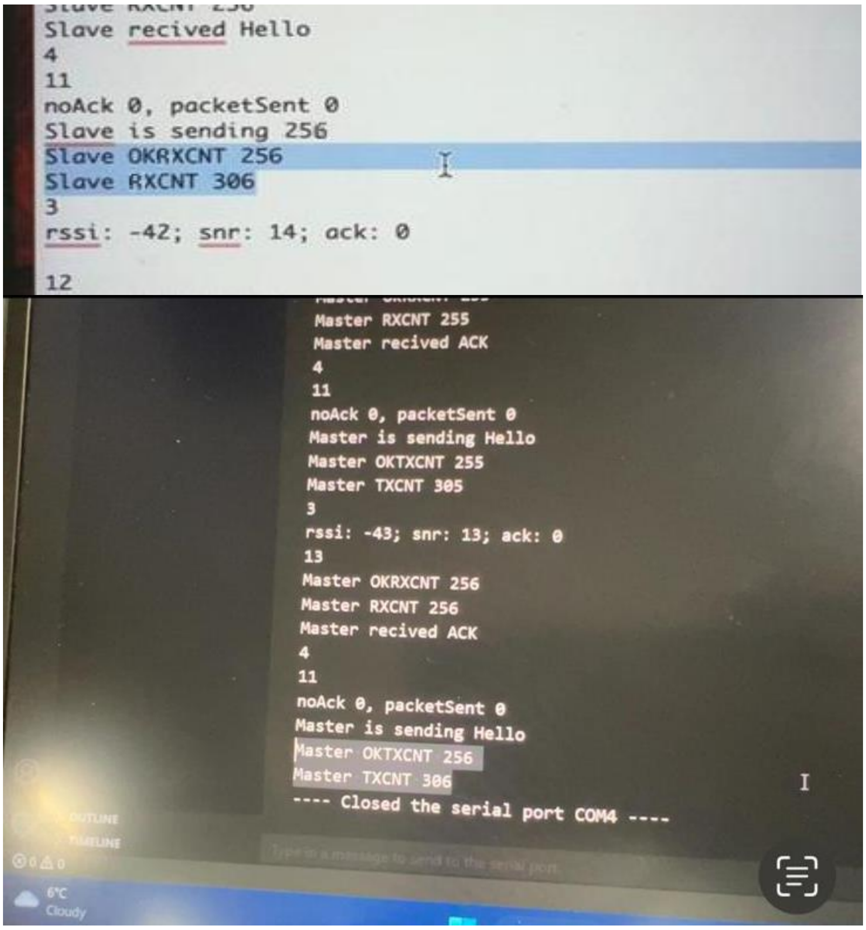


Figure 22 - Test 3 - Packet Matching Algorithm. Packets Lost Only at Slave's End

As it is proven and well explained in the above tests that the Channel Probing and Quantization algorithms work as predicted, it is required now to test the other stages of the key generation process.

Early Version Testing of Information Reconciliation and Privacy Amplification Algorithms

In the fourth test below (Test 4), an early version of the completed key generation algorithm can be viewed on the two serial monitors. It is called an early version because the final version uses a different adaptation of the pseudocode in Figure 9.

1) In the channel probing stage of the algorithm, for each of the two serial monitors, 144 packets are wirelessly sent between Alice and Bob.
2) Then the recorded RSSI values are displayed and a mean value (or average as displayed in the figure) is calculated from these saved values. With the use of the mean, two 144 binary keys are generated separately for Alice and Bob based on the recorded RSSI values and their calculated mean values.
3) The text on the serial monitors between the generated key to the hashed key is the BCH Information Reconciliation algorithm. This algorithm is the exact one as the Pseudocode algorithm presented in Figure 9. Once the Information Reconciliation stage finishes, both keys are the same for Alice and Bob, as in Test 4 in the figure below.
4) Once the keys are the same for the two IoT devices, a hash key is produced. This hash key can be used for checking the validity of the algorithm. As can be viewed in the figure below, both hashed keys are the same. Therefore, the generated keys are the same. So, the key generation algorithm works, and the devices now enter Sleep Mode.

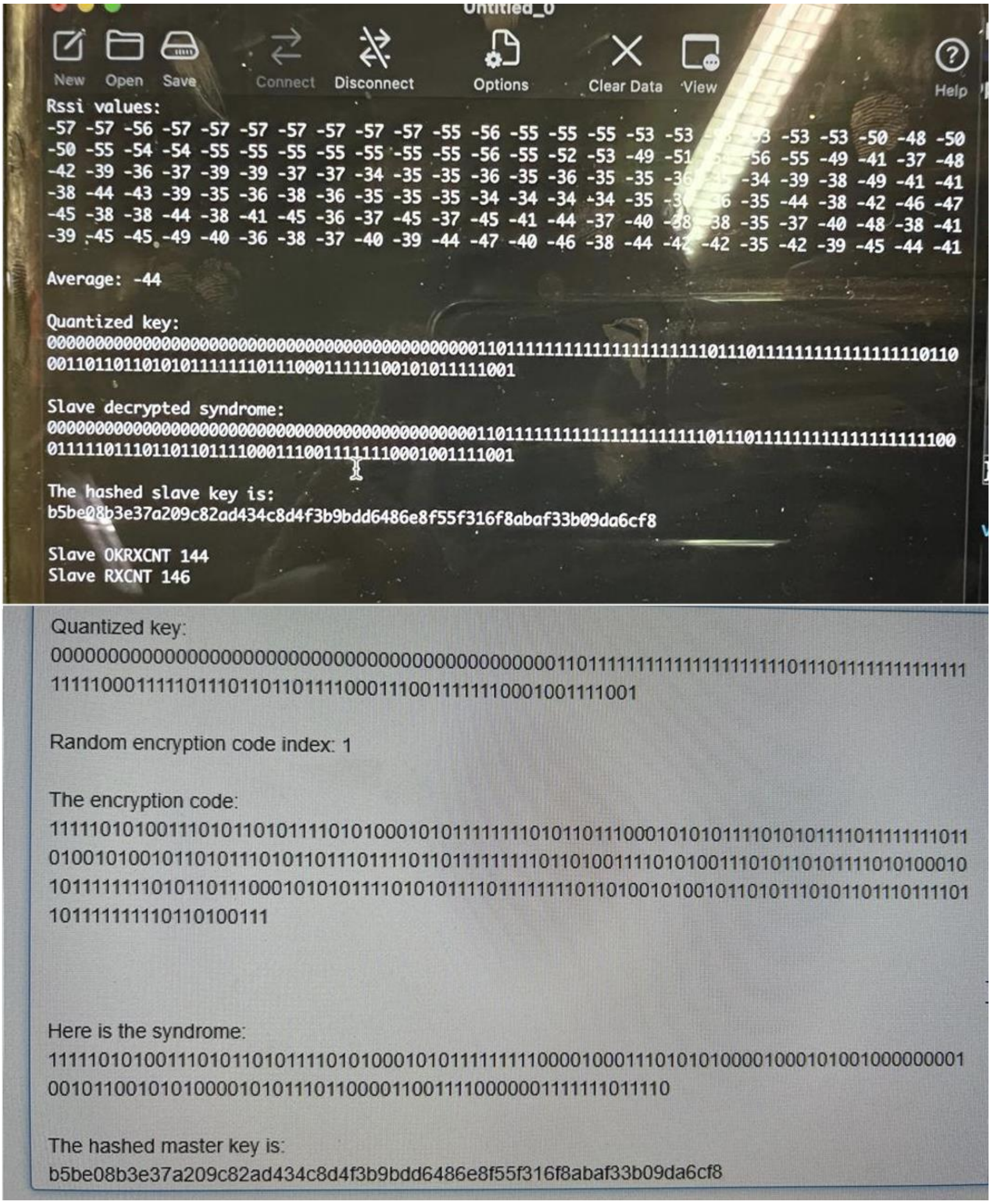


Figure 23 – Test 4 - Early Version Example of Completed Key Generation Process Using BCH Code

Final Version Test for the Key Generation Algorithm

In the image below, a test (Test 5) of the final version of the implemented key generation algorithm can be viewed. The most notable difference between this test and Test 4 is, of course, the information reconciliation algorithm. In both the early and the final versions, a key is generated and hashed for the two devices. Although in this test, codes are more randomly selected to create the random code which is used for encoding the master key and decoding it at the slave's end from the syndrome. As before, for power efficiency, once the algorithms finish and both Alice and Bob have the same key, they go into Sleep Mode.

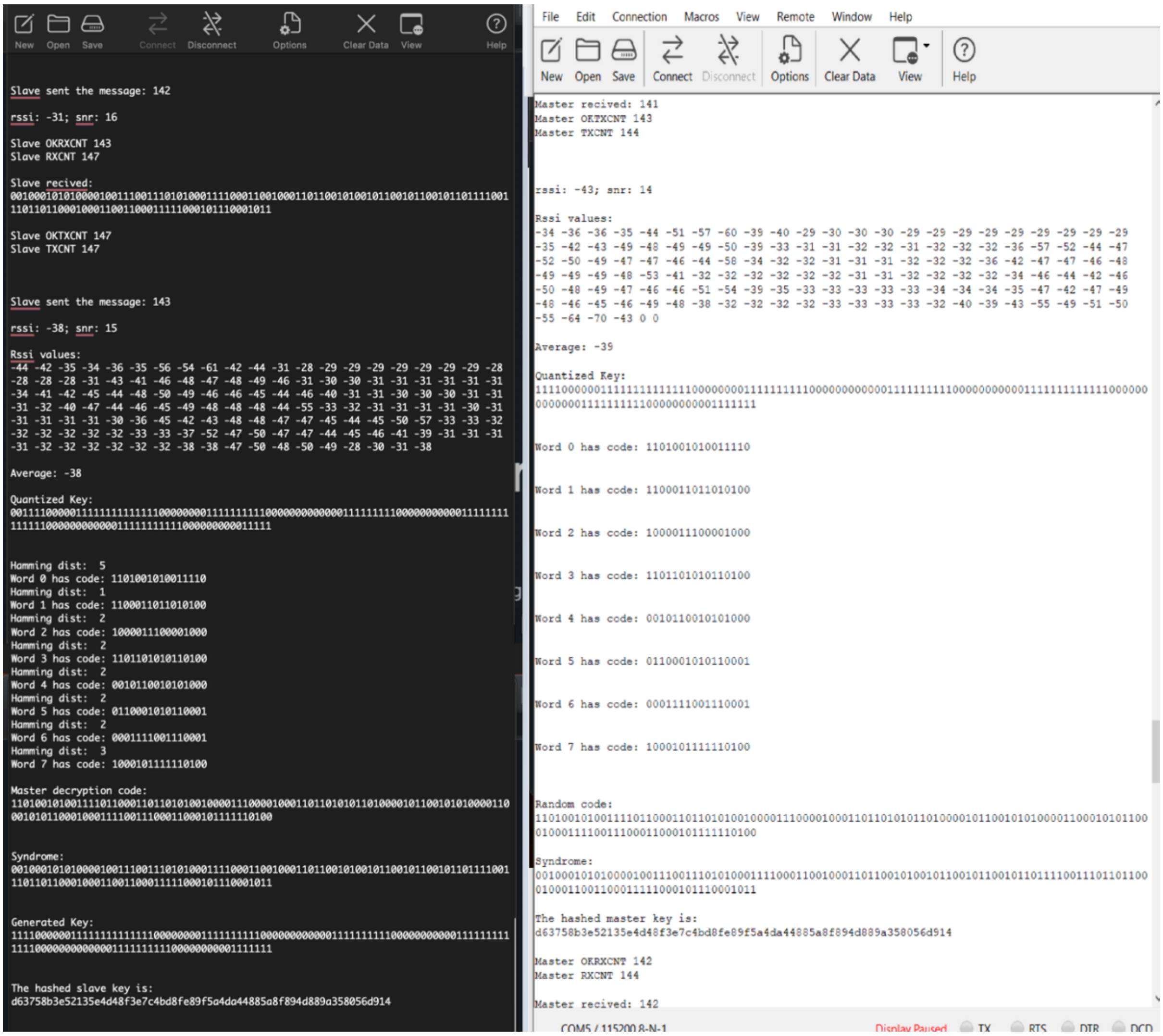


Figure 24 – Test 5 - Example of Final Working Algorithm with Completed Key Generation Process Using Self-Developed Algorithm

Discussion

Reflection on Learning

Conclusions

Abbreviations

IoT Internet of Things
LPWAN Low-Power Wide Area Network
LoRa Long-Range (but also the technology itself)
PHY physical layer
CSS Chirp Spread Spectrum
BW Bandwidth
SNR Signal-to-noise ratio
RSSI Received Signal Strength Indicator
BPSK Binary phase-shift keying

SHA-256 Secure Hash Algorithm 256-bit
MD-5 Message Digest Algorithm
AES Advanced Encryption Standard
UL Uplink
DL Downlink
BCH Code Bose–Chaudhuri–Hocquenghem Code

References

Bibliography

[1] H. R. a. S. G. J. Zhang, “Experimental investigation on Wireless Key Generation for low-power wide-area networks,” *EEE Xplore,* 2019.

[2] W. &. H. M. Diffie, New directions in cryptography, IEEE Transactions on Information Theory, vol. 22, no. 6, pp. 644-654, Nov. 1976, doi: 10.1109/TIT.1976.1055638.

[3] A. v. O. O. &. V. S. Menezes, Handbook of applied cryptography, CRC press, 1996.

[4] N. I. o. S. a. Technology, “Secure Hashing,” 2015. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-53/rev-4/final.

[5] E. U. A. f. N. a. I. Security, “Guidelines on Cryptographic Algorithms and Key Lengths,” 2017. [Online]. Available: https://www.enisa.europa.eu/publications/guidelines-on-cryptographic-algorithms-and-key-lengths-2014/at_download/fullReport.

[6] I. E. T. Force, “Cryptographic Hash Functions,” 2015. [Online]. Available: https://tools.ietf.org/html/rfc6151.

[7] J. Z. X. W. a. L. H. Y. Zou, “A survey on wireless security: Technical challenges recent advances and future trends,” *Proc. IEEE,* vol. 104, no. 9, pp. 1727-1765, 2016.

[8] L. V. A. Z. a. M. Z. S. Centenaro, “Long-range communications in unlicensed bands: the rising stars in the IoT and smart city scenarios,” *IEEE Wireless Communications,* vol. 23.

[9] M. P. M. M. a. L. P. A. Borghetti, “Long Range and Low Power IoT Communication: LoRaWAN,” *Journal of Sensor and Actuator Networks,* vol. 8, no. 3, Sept. 2019.

[10] M. I. J. Z. a. S. G. Henri Ruotsalainen, “Experimental Investigation on Wireless Key Generation for Low Power Wide Area Networks,” https://livrepository.liverpool.ac.uk/3057122/1/IoTJ2019-LoRaWAN-KeyGen.pdf.

[11] L. Alliance, “LoRaWAN 1.1 Specification,” 2018. [Online]. Available: https://resources.lora-alliance.org/technical-specifications.

[12] *. M. L. a. L. W. Xingda Chen, “A Complete Key Management Scheme for LoRaWAN v1.1. doi: 10.3390/s21092962,” National Center for Biotechnology Information, 2021 May.

[13] T. T. Network, “The Things Network,” The Things Network, [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/security/.

[14] B. L. Bram Cohen, “AES-hash,” 2 May 2001. [Online]. Available: https://csrc.nist.rip/groups/ST/toolkit/BCM/documents/proposedmodes/aes-hash/aeshash.pdf.

[15] (. I. S. W. 1. ,. (. I. M. S. 1. ,. (. M. I. B. P. 2. ,. (. M. I. A. K. R. 1. ,. (. I. NUR HAYATI 1, “A Novel Session Key Update Scheme for LoRaWAN,” IEEE Access, 30 August 2022. [Online]. Available: https://shura.shu.ac.uk/30659/1/Pranggono-NovelSessionKey%28VoR%29.pdf.

[16] E. B. a. J. Morrish, “Forecasting the Internet of things revenue opportunity,” *https://machinaresearch.com/report/forecasting-the-internet-of-things-revenue-opportunity,* April 2015.

[17] K. Z. R. X. W. X. a. P. C. X. Xiong, “Low power wide area machine-to-machine networks: Key techniques and prototype,” *IEEE Commun,* vol. 53, no. 9, pp. 64-71, Sept. 2015.

[18] A. K. V, “Embien, INTRODUCTION TO LORA TECHNOLOGY,” 27 February 2019. [Online]. Available: https://www.embien.com/blog/introduction-to-lora-technology/.

[19] P. K. a. M. S.] U. Raza, “Low power wide area networks: An overview,” *IEEE Commun. Surveys Tuts,* vol. 19, no. 2, pp. 855-873, 2017.

[20] S. R. C. T. i. W. Communications, “LoRa: A Low-Power Wide-Area Network for the Internet of Things,” https://heim.xyz/documents/LoRa-SeminarReport-heim-xyz.pdf.

[21] L. e. f. I. connectivity. [Online]. Available: http://resources.alcatel-lucent.com/asset/200178.

[22] SEMTECH, “SX1280/SX1281/SX1282 Long Range, Low Power, 2.4 GHz Transceiver with Ranging Capability, pg. 30,” SEMTECH.

[23] STM, “https://www.st.com/en/evaluation-tools/nucleo-l073rz.html,” STM, [Online]. Available: STM32 Nucleo-64 development board with STM32L073RZ MCU, supports Arduino and ST morpho connectivity.

[24] MBED, “Mbed.com,” OS MBED, 13 March 2017. [Online]. Available: https://os.mbed.com/teams/Semtech/code/SX1280PingPong/diff/22e02d1cfbca/main.cpp.

[25] V. Setti, “Mbed-TLS,” [Online]. Available: https://github.com/Mbed-TLS/mbedtls.

[26] A. Nocturne, “Medium.com Understanding Bitcoin’s Algorithm and Breaking SHA256,” Nerd For Tech, May 2021. [Online]. Available: https://medium.com/nerd-for-tech/understanding-bitcoins-algorithm-and-breaking-sha256-42a636cc9de6.

[27] techtelegraph.co.uk, “techtelegraph.co.uk,” techtelegraph.co.uk, Quantum Computers Still Unable to Crack Bitcoin SHA256 Algo, 10 January 2023. [Online]. Available: https://techtelegraph.co.uk/quantum-computers-still-unable-to-crack-bitcoin-sha256-algo/.

[28] A. M. L. H. Junqing Zhang, “Channel-Envelope Differencing Eliminates Secret Key Correlation: LoRa-Based Key Generation in Low Power Wide Area Networks,” *IEEE Transactions on Vehicular Technology,* vol. 67, no. 12, pp. 12462 - 12466, December 2018.

Appendices