Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea Magistrale in Ingegneria delle Telecomunicazioni

Security threat analysis and countermeasures for LoRaWAN™ join procedure

Laureando

Simone Zulian

Relatore

Prof. Stefano Tomasin

Co-relatore

Prof. Lorenzo Vangelista

Anno Accademico 2015/2016

Abstract

LoRaWAN™ is a new protocol designed to provide Low Power Wide Area Network with features specifically needed to support low-cost, mobile, secure bi-directional communication for Internet of Things (IoT). Indeed, the IoT paradigm may present some very specific features that cannot be easily integrated with the constraints of cellular or other type of existing networks but requires dedicated hardware and networks. Our aim is both to analyze the security threats of join procedure and to supply countermeasures. Moreover we also examine the state of the art implementation of the protocol, focusing the attention on the procedure for the generation of random numbers such the DevNonce. In particular we have theoretically and experimentally verify that, in some situations, the generation of bits (and of DevNonce) can be non-uniform.

Contents

Chapter 1

Introduction

Internet of Things (IoT) is going to take a major place in the telecommunications market as announced in technical and public medias [1]. The paradigm of IoT relies on the deployment of billions of objects having the capability of transmitting information about their context and environment and to create a real-time, secured and efficient interaction between the real and the virtual worlds. IoT revealed to be a key technology for solving societal issues such as digital cities, intelligent transportation, green environment monitoring or medical care and elderly person monitoring.

The main challenge of this new paradigm is to let a very huge number of machine type devices (MTDs) be connected to the Internet at a low cost, with a limited infrastructure and featuring a very long life time with very small battery or energy needs.

In this global picture, there exist different technical issues. M2M has been first defined to connect MTDs in their vicinity. The proposed solutions extensively rely on research results produced over the last twenty years for ad-hoc and wireless sensor networks. Starting twenty years ago from theoretical concepts, this very active research area went up to the definition of full standards (802.15.4, 802.15.6, Zigbee, Bluetooth) which already found a market.

More recently, the IoT paradigm has been extended to the problem of connecting all these MTDs to the Internet, and through Internet to anyone or anything. The massive connection of objects spread over the world is a challenge that has some similarities with the paradigm of cellular networks which aimed at connecting people. This similarity attracted the interest of mobile network providers, to exploit such attractive potential market and IoT has been identified as a target for the future 5G, while several proposals already exist to adapt the 4G technology to IoT. Nevertheless the IoT paradigm may present some very specific features that cannot be easily in-

tegrated with the constraints of cellular networks. In many applications, the individual targeted throughput is very low and the capacity is not a relevant criterion. On the opposite, the latency, the energy efficiency or the reliability are more critical. Except for cars or few other mobile objects, IoT may rely mostly on static nodes. But the dynamic of the problem comes from the fact that these nodes may transmit a packet with a very low probability (e.g. once a week or once a month). Keeping these nodes continuously connected would be not efficient and an important issue is to allow a fast and reliable bursty connection. For these reasons, recently, new network technologies have been deployed. These networks better comply with the specific features of IoT, through dedicated physical and MAC layers.

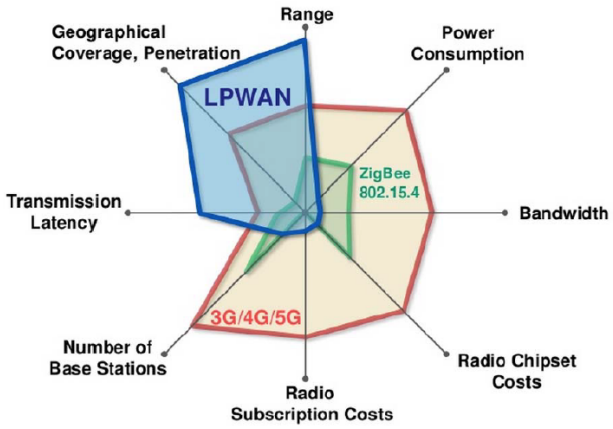

Figure 1.1: Comparison between different paradigms of IoT and cellular networks.

Low Power Wide Area Networks (LPWAN) are an example of networks recently deployed, that allow long range communications at a low bit rate among connected objects, such as sensors operated on a battery. LPWAN technologies include SigFox, LoRaWAN, Adaptrum, WEIGHTLESS, IEEE 802.22, Ingenu and many others.

In particular in this work we focus our attention on LoRaWAN^{TM} protocol. Since it is a new protocol and it is still at an early stage of development, it presents some confusing sections that make way to different interpretations, while other aspects are left to developers. However we also analyze the recommended procedure for the generation of DevNonce for transceiver SX1272. Even if the procedure is not described in the protocol, at the state of the art, every end-device connected to a LoRaWAN network is equipped with SX1272 transceiver, and for the generation of DevNonce (a 16-bit random numbers used in the protocol) it performs the recommended procedure.

So we structure the thesis as described below.

In *chapter 2* we briefly describe the terminology used in LoRaWAN networks. Then we introduce the protocol with particular attention on security mechanisms and join procedure. Finally we present a survey about discovered LoRaWAN problems.

In *chapter 3* we firstly analyze the security mechanisms used on the join procedure. Later we study what is the best implementation of the protocol (considering what is not specify by the protocol). Finally we observe if there are security breaches on the procedure and what can be changed.

In *chapter 4* we describe a general model of a superheterodyne receiver (such as SX1272), and then we integrate the model with the features of SX1272 focusing the attention on the hardware that permits to write the RSSI values on a register.

Chapter 5 is dedicated to random number generators (RNGs). In the first part we describe what is necessary in order to have a good random number generator. Then we analyze the procedure recommended for the random number generation with SX1272 and, later, we introduce how, theoretically, this procedure can be dangerous and not efficient.

In *chapter 6* we analyze experimentally if the SX1272 is equipped with a good random generator, or if, in some cases, the RNG is not efficient.

Finally in *chapter 7* we sum up the work of the thesis, analyzing what is the contribution of the thesis and future research directions.

Chapter 2

LoRaWAN^{TM}protocol

LoRa® is a modulation technique that is based on spread-spectrum techniques and a variation of chirp spread spectrum (CSS) with integrated forward error correction (FEC). LoRa significantly improves the receiver sensitivity and uses the entire channel bandwidth to broadcast a signal, making it robust to channel noise and insensitive to frequency offset. The LoRa modulation is the physical layer which can be utilized by many different protocol architectures, such as Mesh, Star, 6lowPAN, etc [2].

LoRaWAN™ is a MAC protocol for a high capacity, long range star network that the LoRa Alliance has standardized for Low Power Wide Area Networks (LPWAN).The LoRaWAN protocol is optimized for low cost battery operated sensors and includes different classes of nodes to optimize the trade-off between network latency and battery lifetime. LoRaWAN is deployed for nationwide networks by major telecom operators, in order to make sure the different nationwide networks are interoperable [2].

LoRaWAN networks typically are laid out in a star-of-stars topology in which **gateways** relay messages between **end-devices** and a central **network server** at the backend. Gateways are connected to the network server via standard IP connections while end-devices use single-hop LoRa^{TM}or FSK communication to one or many gateways. Communication is generally bidirectional, although uplink communication from an end-device to the network server is expected to be the predominant traffic.

LoRa endpoints are the elements of the LoRa network where sensing or control is undertaken. They are normally remotely located and battery operated.[2].

The **LoRa gateways** are multi-channel, multi-modem transceivers that can demodulate on multiple channels simultaneously and even demodulate multiple signals on the same channel simultaneously due to the properties of LoRa. The gateways use different radio frequencies components than the

end-point to enable high capacity and serve as a transparent bridge relaying messages between end-devices and a central network server in the backend. [2]

[4] The **LoRa network server** manages the network. The network server acts to eliminate duplicate packets, schedules acknowledgement, and adapts data rates. In view of the way in which it can be deployed and connected, makes it very easy to deploy a LoRa network. At the state of the art the Network Server is not so well defined by the standard. However in several systems already deployed the Network Server is an Internet facing web service which the Gateways can connect to using for instance cellular networks [5].

A remote computer can then control the actions of the endpoints or collect data from them (Fig. 2.1).

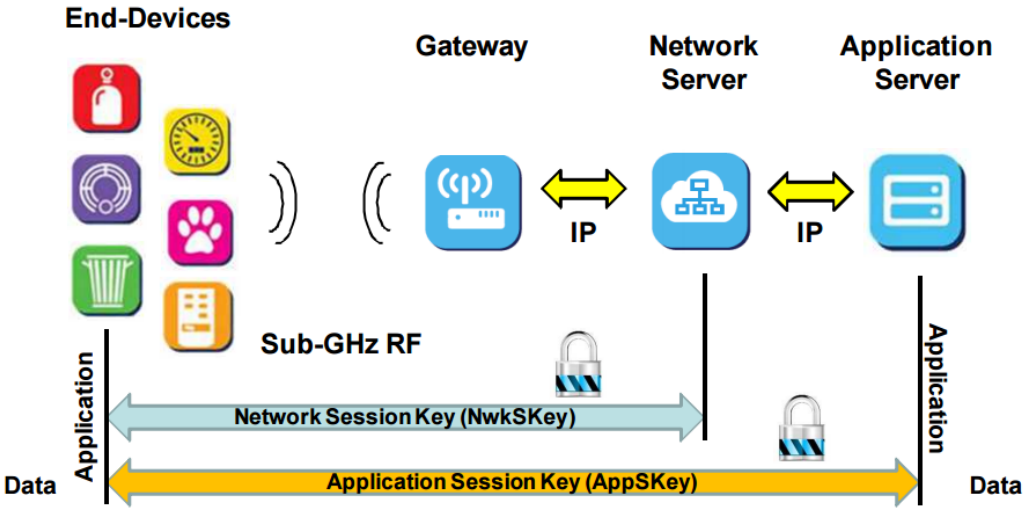

Figure 2.1: LoRa network architecture.

Communication between end-devices and gateways is spread out on different frequency channels and data rates. The selection of the data rate is a trade-off between communication range and message duration and it ranges from 0.3 kbps to 50 kbps. Communications with different data rates do not interfere with each other. To maximize both battery life of the end-devices and overall network capacity, the LoRa network infrastructure can manage the data rate and radio frequency output for each end-device individually by means of an adaptive data rate (ADR) scheme.

End-devices may transmit on any available channel at any time, using any available data rate, as long as the following rules are respected:

- The end-device changes channel in a pseudo-random fashion for every transmission. The resulting frequency diversity makes system more

robust to interference.

- The end-device respects the maximum transmit duty cycle relative to the sub-band used and local regulations.
- The end-device respects the maximum transmit duration (or dwell time) relative to the sub-band used and local regulations [3, Ch.1].

LoRaWAN Classes All LoRaWAN devices implement at least the Class A functionality. In addition they may implement options named Class B and Class C. End-devices of Class A allow for bi-directional communications whereby each end-device's uplink transmission slot scheduled by the end-device is based on its own communication needs with a small variation based on a random time basis (ALOHA-type protocol). This Class A operation is the lowest power end-device system for applications that only require downlink communication from the server shortly after the end-device has sent an uplink transmission. Downlink communications from the server at any other time will have to wait until the next scheduled uplink.

End-devices of Class B allow for more receive slots. In addition to the Class A random receive windows, Class B devices open extra receive windows at scheduled times. In order for the End-device to open it receive window at the scheduled time it receives a time synchronized Beacon from the gateway. This allows the server to know when the end-device is listening.

End-devices of Class C have nearly continuously open receive window, only closed when transmitting. Class C end-device will use more power to operate than Class A but they offer lowest latency for server to end-device communication.

2.1 End-Device cryptography and commissioning

LoRaWAN protocol expects that all payloads are encrypted using an AES algorithm, described in IEEE 802.15.4/2006 Annex B [IEEE802154], using a 128 bits secret key, that is the Application Session Key (AppSKey) if the payload carries data information and the Network Session Key (NwkSKey) if the payload carries MAC messages. Furthermore all frames contain a 32 bits cryptographic MIC signature computed using the NwkSKey over the entire frame and described in [RFC4493] (Fig. 2.2).

AppSKey must be only known by end-device and application server; NwkSKey, instead, must be known by end-device and network server only.

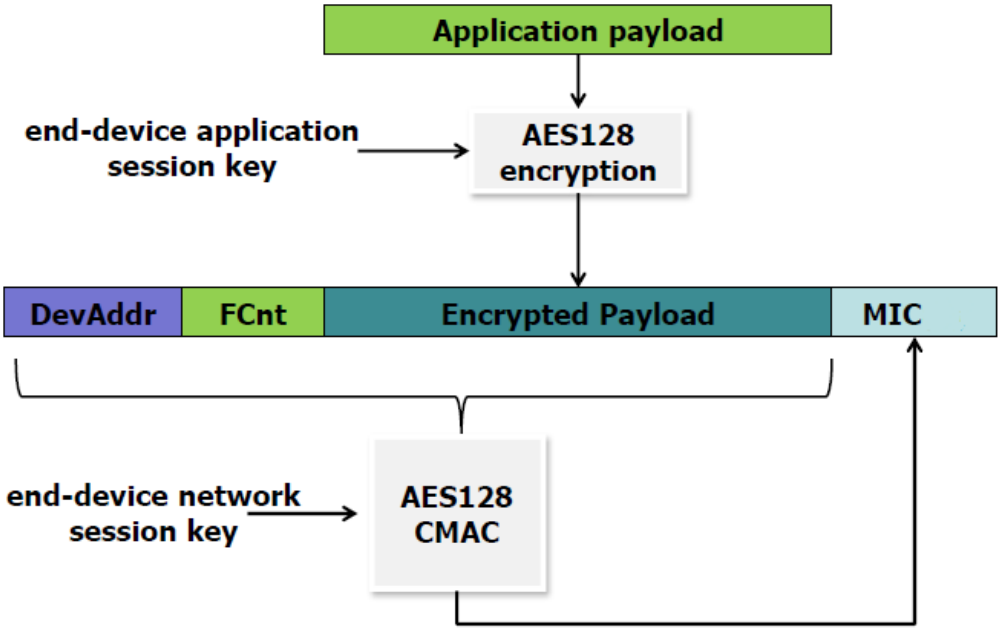

Figure 2.2: Encryption of payload and message signature.

The only components of the network involved are the end-device, the network server and the application server. The gateways, instead, are totally transparent from a security perspective.

Upon reception of a frame, the network server checks that the frame received MIC signature matches the one computed using the end-device's network session key contained in its key database. If the two MICs match then the frame is really coming from legitimate end-device and its content hasn't been modified in any way. The same process happens on the downlink messages (from network server to the end-device). Because each frame contains a frame counter, used also for the evaluation of MIC, the replay attacks are forbidden.

2.1.1 End-device activation

[3, Ch.6] To participate in a LoRaWAN network, each end-device has to be personalized and activated. The activation can be achieved in two ways, either via **Over-The-Air-Activation** (OTAA) when an end-device is deployed or reset, or via **Activation By Personalization** (ABP) in which the two steps of personalization and activation are done as one step.

After activation the end-device stores the following information:

Device address (DevAddr). It consists of 32 bits that identifies the end-device within the network, and it has the following format

Bit#	[31..25]	[24..0]
DevAddr bits	NwkID	NwkAddr

where the most significant 7 bits are used as network identifier (NwkID) to separate addresses of territorially overlapping networks of different network operators and to remedy roaming issues. The least significant 25 bits, the network address (NwkAddr) of the end-device, can be arbitrarily assigned by network manager.

Application identifier (AppEUI). It is a global ID in IEEE EUI64 address space that uniquely identifies the application provider of the end device. The AppEUI is stored in the end-device before the activation procedure is executed.

Network session key (NwkSKey). It is a network session key with length of 128 bits specific for the end-device. It is used by both the network server and the end-device to calculate and verify the **MIC** (Message Integrity Code) of all data messages to ensure data integrity. It is further used to encrypt and decrypt the payload field of a MAC-only messages.

Application session key (AppSKey). It is an application session key with length of 128 bits specific for the end-device. It is used by both the network server and the end-device to encrypt and decrypt the payload field of application-specific data messages. It is also used to calculate and verify an application-level **MIC** that may be included in the payload of application-specific data messages (if the layers above LoRaWAN provide pre-encrypted frame payload).

Over-the-Air Activation For over-the-air activation, end-devices must follow a join procedure prior to participating in data exchanges with the network server. An end-device has to go through a new join procedure every time it has lost the session context information.

The join procedure requires the end-device to be personalized with the following information before it starts the join procedure:

End-device identifier (DevEUI). It is a global end-device ID in IEEE EUI64 address space that uniquely identifies the end-device.

Application identifier (AppEUI). It has been described above.

Application key (AppKey). It is an AES-128 application key specific for the end-device that is assigned by the application owner to the end-device and most likely derived from an applicaion-specific root key exclusively known to and under the control of the application provider. Whenever an end-device joins a network via over-the-air activation, the AppKey is used to derive the session keys NwkSKey and AppSKey specific for that end-device to encrypt and verify network communication and application data.

The join procedure consists of two messages exchanged between end-device and network server, namely *join request* and *join accept*. The first message, the *join request* message, is sent by the end-device to the network server and it has the following format

Size (bytes)	8	8	2
Join Request	AppEUI	DevEUI	DevNonce

It consists of the AppEUI, devEUI and a nonce of 16 bits (**DevNonce**). The DevNonce is a random value. For each end-device, the network server keeps track of a certain number of DevNonce values used by the end-device in the past, and ignores join request with any of these DevNonce values from that end-device. In this manner it is possible to prevents replay attacks by sending-previously recorded join-request messages with the intention of disconnecting the respective end-device from the network. As for all the MAC commands sent as a separate data frame, the message is contained in the frame payload of the LoRa message with port field set to 0. The MIC value for a join request message is calculated as follows:

$cmac$ = aes128_cmac(AppKey,MHDR|AppEUI|DevEUI|DevNonce)

MIC = $cmac$[0..3].

where the notation for the byte is little endian. Moreover the join-request message is not encrypted.

The network server will respond to the join-request message with a join-accept message if the end-device is permitted to join a network, instead no response is given to the end-device if the join request is not accepted. The message has the following format:

Size (bytes)	3	3	4	1	1	(16) Optional
Join Accept	AppNonce	NetID	DevAddr	DLSettings	RxDelay	CFList

The RxDelay is the delay that the end device has to wait between the transmission of the packet and the start of the first receive window. The CFList is an optional list of channel frequencies for the network the end-device is joining. The DLsettings field contains the downlink configuration

Bits	7	6:4	3:0
DLsettings	RFU	RX1DRoffset	RX2 Data rate

where The RX1DRoffset field sets the offset between the uplink data rate and the downlink data rate used to communicate with the end-device on the first reception slot (RX1) and RX2 Data Rate is the data rate of the second receive window.

The application nonce (**AppNonce**) is a 24 bits random value or some form of unique ID provided by the network server and used by the end-device to derive the two session keys NwkSKey and AppSKey as follows:

NwkSKey = aes128_encrypt(AppKey,0x01|AppNonce|NetID|DevNonce|pad_{16})
AppSKey = aes128_encrypt(AppKey,0x02|AppNonce|NetID|DevNonce|pad_{16})

where pad_{16} function appends zero octets so that the length of the data is a multiple of 16 bytes. Instead the MIC value for a join-accept message is calculated as follows:

$cmac$ = aes128_cmac(AppKey,MHDR|AppNonce|NetID|DevAddr|
DLSettings|RxDelay|CFList)
MIC = $cmac$[0..3].

The join-accept message itself is encrypted with the AppKey as follows:

aes128_decrypt(AppKey,MHDR|AppNonce|NetID|DevAddr|DLSettings|
RxDelay|CFList|MIC).

Activation by Personalization (ABP) Activation by personalization directly ties an end-device to a specific network by-passing the join procedure. Activating an end-device by personalization means that the DevAddr and the two session keys NwkSKey and AppSKey are directly stored into the end-device instead of the DevEUI, AppEUI and the AppKey. The end-device is equipped with the required information for participating in a specific LoRa network when started.

Each device should have a unique set of NwkSKey and AppSKey. Compromising the keys of one device shouldn't compromise the security of the communications of other devices. The process to build those keys should be such that the keys cannot be derived in any way from publicly available information (like the node address for example).

2.2 Literature about LoRaWAN problems

Since LoRaWAN is a recent protocol, many aspects are not so clearly or well defined. Moreover some features seem to be critical from a security perspective.

MWR Labs in [5] are pointed out some of these security problems. It should be possible to use LoRa solutions securely to protect against man in the middle attacks affecting the confidentiality and integrity of data. LoRa also provides ways for developers to securely add new nodes to their LoRa network. However other areas are left to the developers, which may lead to security vulnerabilities being introduced into particular LoRa instances.

For nodes, they should only be storing keys that they require. It is likely given the range of hardware attacks available that an attacker could recover the AppKey, NwkSKey and AppSKey from a node using for example side channel analysis. This attack uses the variations in power consumption or EM emissions from the transceiver during AES encryption to determine the key that must have been used. As an attacker with this key would be able to produce correctly signed and encrypted messages, the data coming from individual nodes should therefore be assumed to be potentially untrustworthy. Moreover, the tampering of a device, cloning its AppKey is more dangerous because it is not possible, or very difficult, to change the AppKey of a device.

There exists another issue if the LoRa node used a transceiver (such as the RN2483 of Microchip), which handles encoding, encrypting and transmitting the LoRa data. The microcontroller does not know the encryption keys used by the LoRa network. Instead it would send data to the LoRa transceiver module which would encrypt, sign and transmit the data. An attacker with physical access to one of these devices could in theory replace the microcontroller or use the UART pins of the LoRa transceiver to start sending their own messages on behalf of the node.

Moreover many solutions have made some components Internet facing, e.g. they can be accessed by anyone who knows the IP address, port and protocol that they use. Some LoRa solutions have made their Network Servers Internet facing so that they can be connected to by the Gateways. This increases the risk of compromise, as Internet facing services are a common target for hackers. One risk is that gateway traffic could now be forged without the need of a compromised node and therefore forgo the cost, as well as the bandwidth limitations that this vector causes. A possible attack would be for the MIC of packets to be brute forced (which would take in average around 2 billions attempts to succeed given the MIC's 4 byte key space). Although infeasible over LoRa, a web service could be sent this amount of traffic. In order to prevent any possible attack the link between application

and network server must be secure (SSL) and authenticated (certificate).

Another company, Gemalto, evidences a conflict of interest in the provisioning of the keys[11]. Indeed in OTAA the same AppKey is used to derive both the NwkSKey and the AppSKey, so the network operator is able to read the application data and the application server must trust the network server to not modify the payloads (since it can calculates the AppSKey). Instead the application provider, knowing the NwkSKey, can clone devices. For this reason Gemalto proposes the employment of a trusted third party for the generation of keys.

Another critical aspect is the interference between adjacent networks. LoRaWAN is a lossy protocol, due to its uncoordinated, asynchronous nature. If multiple LoRaWAN networks are present in the area, additional interference will increase packet-error-rate. Since all LoRaWAN channels are shared, any LoRaWAN packet is seen and demodulated by all gateways in range, no matter who owns them. If there is a carrier operated LoRaWAN network and several private LoRaWAN networks operating in an area, performance of all networks will suffer due to collisions. Also, since LoRa has a low co-channel dynamic range, without a closed-loop power control scheme, any nodes close to the gateway will drown out nodes far away[12][13]. These issues can be exploited to perform a DoS attack.

To the best of my knowledge, a feature of the protocol has been inadequately analyzed: the join procedure and the use of DevNonce field. For this reason in the next chapters we analyze the procedure focusing in what aspects are dangerous, hypothesizing also an incorrect implementation of the protocol, due to a not sufficiently clear description of the process.

Chapter 3

Join procedure analysis

3.1 Security mechanisms of join procedure

The integrity, authenticity and non copy of join procedure messages (join request and join accept) is guaranteed by:

- MIC field of message;
- 16 bits random number called DevNonce.

The MIC field permits to sign the message and guarantee its integrity, because it is evaluated through the fields of the message (included the DevNonce) and the AppKey, known only by that end-device and by the server. A malicious node that wants to modify the message or pretend to be the legitimate node, is not able to calculate a valid MIC since it doesn't know the AppKey, and its messages is discarded by the network server.

The DevNonce, instead, has been introduced in the join procedure in order to avoid replay attacks. Let's consider an example of situation represented in Fig. 3.1 where they are present an end-device A that has to join the network and a malicious node M. If the node M has registered the join request message of A, and it sends the message after an interval T of time, trying to disconnect node A from the network, it fails, because the network server stores a predefined DevNonce values used in the past by node A and rejects join request with DevEUI and AppEUI of node A with past values of DevNonce. Without the introduction of this mechanism, the node M is able to disconnect node A from the network. Indeed at every join request, new session keys are generated (AppSKey and NetSKey). If the server received a previously recorded join request message (that contains a valid MIC) but it isn't able to distinguish that the message is a replica, it responds with an encrypted join accept message with the parameters used to evaluate the new

session keys. So, after this moment, the legitimate node A sends messages encrypted with the old session keys, while the server decrypts messages using the new session keys, and vice versa.

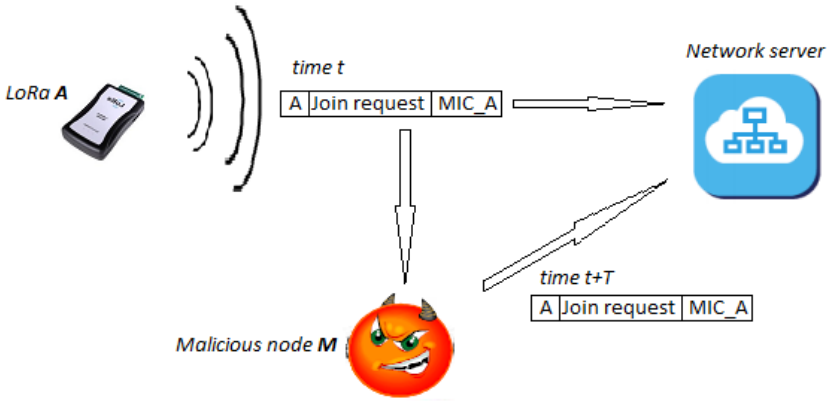

Figure 3.1: Example of replay attack.

3.2 Join procedure problems

In spite of these mechanisms the join procedure presents some critical aspects. In particular the LoRaWAN protocol:

1. doesn't specify the numbers of DevNonce that must be registered by the network server for each end device;

2. briefly and inadequately explains the policy of the network server in the case of join request with previously used DevNonce;

3. doesn't implement a mechanism that prevents replay attacks with the join accept message.

We firstly analyze the first and second aspect that concern with join request message and then we focus our attention on the third point that regards the join accept message.

3.2.1 Problem with join request message

First of all we have to consider the frequency of join procedure per each end device and how many join procedures, in average, each end device performs in its life. We call f_J the number of valid join procedures per day per each

end device, N_D the number of previously used values of DevNonce stored by the network server and T_r the time (in days) that a malicious node has to wait in order to perform a replay attack, that is realized sending a previously recorded join request message with a value of DevNonce not yet stored by the network server. The relationship between these three quantities is

$$T_r[days] = \frac{N_D + 1[DevNonce]}{f_J[DevNonce/days]} \tag{3.1}$$

Obviously each end device owner prefers that T_r is, as possible, potentially infinity. In order to reach this goal, assuming that f_J is an unchangeable quantity, the bigger is the number of stored DevNonce per each end device the larger is T_r. Since DevNonce is a 16 bits integer, the maximum value of N_D is 2^{16} and $T_r \leq 2^{16}/f_J$. However N_D influences also the performance of the join procedure. In particular, depending on the policy adopted in the case of join request with previously used DevNonce, we have two types of malfunctioning:

a) If the network server rejects join request messages with previously used DevNonce, a legitimate request with already used DevNonce is discarded;

b) If the network server switches off the end device that generates a request with previously used DevNonce, that end device can't work anymore.

Even if the LoRaWAN protocol specifies that requests with already used DevNonce (by the same end device) must be drop, some commercial devices, instead of only drop invalid join request, switch off the node that has generated the invalid join request, probably hypothesizing a malfunctioning of the device. Let's analyze the probability of the two events.

Case a) If the network server only drops the requests with already used DevNonce, we are interesting in the probability of generating a stored - DevNonce given N_D. Supposing to have a true random number generator (that is the value of DevNonce has discrete uniform pmf in the alphabet $[1, \ldots, N = 2^{16}]$), and called S the set of stored DevNonce, with $|S| = N_D$ the probability is

$$Pr[dev_K \in S] = \frac{N_D}{N} \tag{3.2}$$

We can observe that this probability is higher if N_D is larger. So there exists a trade-off between the probability in (3.2) and T_r.

In (3.2) we have not considered that, when an end device join a network for the first time, $|S| = N_S < N_D$, that is the number of stored DevNonce

is lower than N_D. However in this case the equation in (3.2) still holds with N_S instead of N_D.

Case b) If the network server implements the policy in which the end device is switched off if a join request with a stored DevNonce arrived, it is important to evaluate the probability to be turned off within a certain amount of time. Let's call dev_k the devNonce generated at the k^{th} join procedure, the probability of generating K different values of devNonce is

$$\begin{aligned} &Pr[\text{K different devNonce}] = Pr[dev_2 \notin \{dev_1\}]\cdot \\ &\cdot Pr[dev_3 \notin \{dev_1, dev_2\}] \cdot \ldots \cdot Pr[dev_K \notin \{dev_1, \ldots, dev_{K-1}\}] = \\ &= \frac{N-1}{N} \cdot \frac{N-2}{N} \cdot \ldots \cdot \frac{N-K+1}{N} = \prod_{i=1}^{K-1} \frac{N-i}{N} = \\ &= \frac{D_{n,k}(N,K)}{N^k} = \frac{N!}{(N-K)!N^K} = \mathcal{D}(K) \end{aligned} \tag{3.3}$$

The probability that at the K^{th} join procedure we generate a DevNonce equal to a previous value is

$$\begin{aligned} &Pr[dev_K \in \{dev_1, \ldots, dev_{K-1}\} \bigcap dev_1 \neq dev_2 \neq \ldots \neq dev_{K-1}] = \\ &= Pr[\text{K-1 different devNonce}] \cdot Pr[dev_K \in \{dev_1, \ldots, dev_{K-1}\}] = \\ &= \mathcal{D}(K-1) \cdot \frac{K-1}{N} = \mathcal{E}(K) \end{aligned} \tag{3.4}$$

The probability in (3.4) corresponds to the probability to be switched off at K^{th} attempt if $K \leq N_D + 1$. If we consider also the case with $K > N_D + 1$ the probability is

$$\begin{aligned} &Pr[\text{node is switched off at K}|N_D] = \\ &= \mathcal{S}(K) = \begin{cases} \mathcal{E}(K) & \text{if } K \leq N_D + 1 \\ \mathcal{E}(N_D+1)\left(1 - \frac{N_D}{N}\right)^{(K-1-N_D)} & \text{if } K > N_D + 1 \end{cases} \end{aligned} \tag{3.5}$$

Then the probability to be switched off within T attempts given N_D is

$$Pr[\text{node is turn off within K attempts}|N_D] = P_{off} = \text{cdf}(\mathcal{S}(K)) \tag{3.6}$$

that is the probability we were interesting in.

Let's now consider typical values of f_J. In order to guarantee a correct operation of the network, most societies prefers to refresh the session keys every day, that is at least 1 valid join procedure per day is performed by each

end device. Considering that in a Lora network an end device is designed to work for 10 years, every end device in its life generates at least $10 \times 365 = 3650$ values of DevNonce. To be confident we multiply this value by a factor of 2 and we assume that an end device in its life generates $3650 \times 2 = 7300 = N_D^{max}$ DevNonce.

Let's firstly analyze the case a). Let's suppose that the network server decides to store $N_D < N_D^{max}$, so by (3.1) $T_r < 10$ years. Let's also assume that it is employing the policy of dropping the invalid requests. What happens if a malicious node M performs a replay attack sent a previously recorded join request after T_r days? Apparently this is not a big problem. Indeed supposing that the legitimate end device A has generated $N_D + 1$ DevNonce from its origin, the first value of DevNonce generated and used by the device is not been stored by the network server yet. Let's consider also that M has registered all the requests sending in these T_r days. Then M is able to disconnect node A from the network using the first replayed message. Let's notice that the server, considering this request as valid, stores the value of DevNonce. This cannot be seen as a big problem if the node A becomes aware to be disconnected to the network and it performs a new (valid) join request. However after this moment $N_D + 3$ valid DevNonce has been generated from the beginning. Supposing that the last generated DevNonce is different from the second generated DevNonce, now M has two registered messages that can be used to disconnect A (the second and the third). Even considering the unlikely case that the last generated DevNonce by A is equal to the second generated DevNonce, M has can send the third generated and registered join request message. So potentially, after T_r, M is able to disconnect A from the network for ever.

Obviously this analysis doesn't consider what is the effective benefit that M can have to wait for T_r days before disconnecting A. In practice T_r can be of the order of years and the advantages of node M to perform this attack can't exist. However theoretically if T_r is lower than the life time of an end device, that end device can't work anymore after T_r days. So it seems to be better to store all the previously used values of DevNonce for each end device. But in this last case, considering $N_D = N_D^{max}$ the probability of generating a previously used DevNonce, using (3.2), is

$$Pr[dev_K \in S] = \frac{N_D^{max}}{2^{16}} = \frac{7300}{65536} \simeq 0.11. \tag{3.7}$$

So it is relatively probable to generate an already stored used DevNonce in the last days of life of an end device. However, dropping the requests, this is not a big issue. Indeed the device will send a join request until a valid DevNonce is created. So the choice of using $N_D \geq N_D^{max}$ should be the best

solution.

More critical is the situation in case b), that is the network server switches off the nodes that generate invalid join request. First of all we consider the plot in Fig. 3.2 where we reported the probability in (3.6) for different values of N_D. We can notice that the larger is N_D the higher is the probability to be switched off earlier. So it seems to be better to choose a low value of N_D. However the reasoning in case a) is still valid in case b). So in order to prevent that type of DoS attack, we must have $N_D \geq N_D^{max}$. But, also considering $N_D = N = 2^{16}$, if we evaluate the average value in (3.4), we have

$$E\left[\mathcal{E}(K)\right] = \sum_{k=1}^{N} k\mathcal{E}(k) \simeq 319.5 \tag{3.8}$$

and this means that in average after 320 join procedures is generated a previously used value of DevNonce. The value obtained differs of one order of magnitude with respect of N_D^{max}. Moreover, from Fig. 3.3, representing the probability that a node is switched off after K generation of DevNonce, considering $N_D = N = 2^{16}$, we can notice how rapidly the probability to be turn off increase. For example the probability to be switched off within a year, that is within $K = 365 \times 2 = 730$, is

$$Pr[\text{node is switched off within a year}] \simeq 0.98$$

Moreover another DoS attack is easy to implement if the policy of the network server is to switch off the end devices that produce invalid join requests. Indeed if the malicious node M registers all the join requests of end-devices around it and sends them to the network server after a period, the network server responds switching off all the nodes for which a replayed join request has been sent. Considering that M can potentially registers the join request messages of nodes located in a range of kilometers and the number of these nodes can be elevated, this problem is potentially catastrophic for a LoRa network.

In conclusion, at the state of the art, considering the presence of these issues the network server must drop the invalid join requests, without switching off the node. Moreover the value of N_D must be as higher as the estimate number of generated and valid join procedure in the life of an end-device. However in the next section, we study if considering a different implementation of the protocol, we achieve higher security.

3.2.2 Problem with join accept message

If for join request message the DevNonce has been introduced in order to prevent replay attacks, for the join accept message a mechanism that prevents

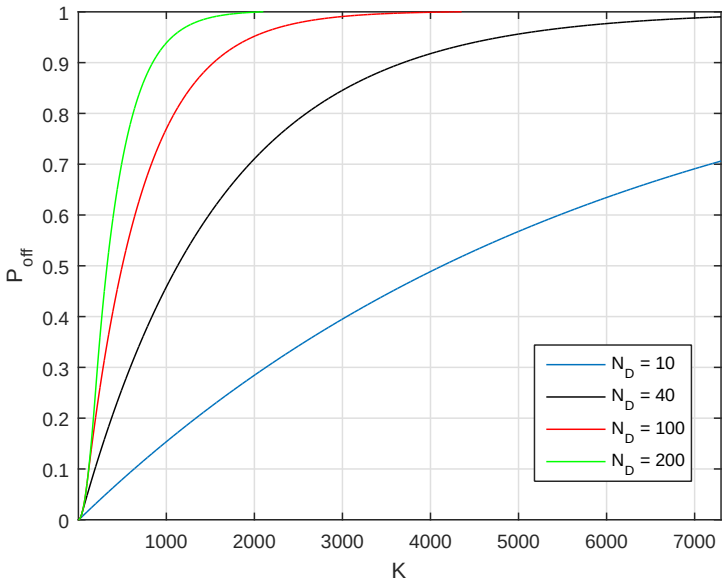

Figure 3.2: Probability to be switched off within K attempts.

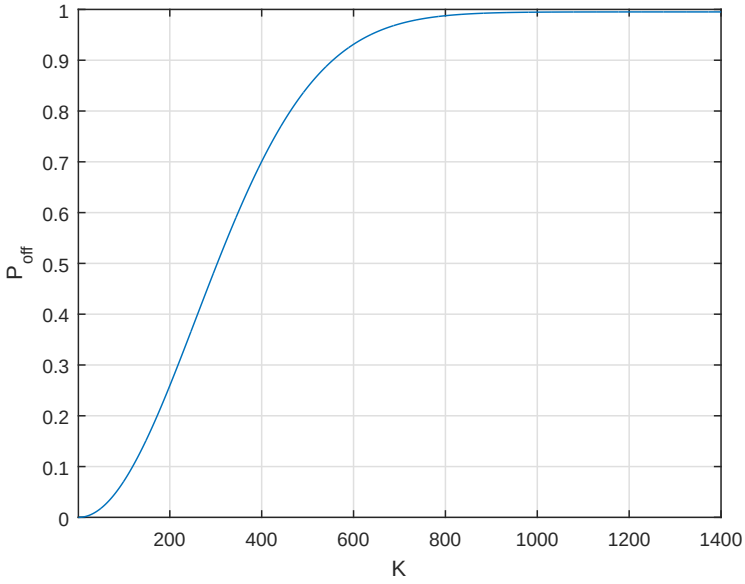

Figure 3.3: Probability to be switched off within K attempts with $N_D = N_D^{max}$.

replay attacks seems to be lacking. Indeed the join accept message includes the AppNonce, a 24 bits random number or unique ID that identify every

message and that it is used to evaluate the session keys. The end-device, however, doesn't register the previously received value of AppNonce (as for DevNonce) and so the AppNonce can't be used as the DevNonce, that is the end device doesn't reject the messages with previously used AppNonce and it can't distinguish between registered messages and correct join accept messages sent by the server. In other words, it is possible to perform a substitution attack. Let's recall the situation described above, where there is a node A sending a join request message and a malicious node M that is able to register the messages sent and received by end-device. Let's suppose that at Day 1 A sends the join request message, called 'jr1', containing 'DevNonce1'. Since 'DevNonce1' is a valid value, the network server responds sending a join accept message, called 'ja1', containing 'AppNonce1', that is registered by M. At day 2 A performs a new join procedure, sending 'jr2' that contains the valid 'DevNonce2'. Then the network server sends 'ja2' containing 'AppNonce2'≠'AppNonce1' (with probability $\frac{2^{24}-1}{2^{24}}$). Let's suppose that M is able to hide 'ja2' to A and send the message 'ja1' in the expected slot of time for the join accept messages. Then A receives the parameters contained in 'ja1' that are 'AppNonce1', 'NetID1', 'DevAddr1', 'DLSettings1', 'RxDelay1' and 'CFList1'. So A uses these parameters to evaluate the two session keys. Instead the network server has used the values contained in 'ja2' and since at least 'AppNonce2'≠'AppNonce1', network server and end device have different session keys. Moreover the day2 'NetID1' and/or 'DevAddr1' may have been assigned to another end device and the network can have two different end device with the same NetID and/or DevAddr.

3.3 Solutions and alternative applications

We propose an easy solution to the substitution attack of join accept message and we analyze two different applications of the DevNonce in the join request message.

3.3.1 Join accept message

A costly solution can be that of adding the last generated DevNonce in the join accept message. Since the network server replies with a join accept message only if a valid DevNonce arrives, the DevNonce in join accept message is unique, i.e. it is different from the previously used DevNonce and the attack described beforehand is not possible. Moreover the end device records the last used DevNonce and accepts the join accept message only if it contains the valid DevNonce. This solution is costly because we have to add 2 octets

to the join accept message. However an ack or another type of message that confirms the procedure is succeeded is not necessary since the end device sends another join request if the procedure fails.

3.3.2 Join request message

Considering the problems highlighted in the previous section we consider two alternative applications:

1. DevNonce is a sequential number;
2. DevNonce is still random but the size is incremented to 24 or 32 bits.

The first proposal is derived analyzing the necessity (or not) for the DevNonce to be random. Indeed it seems to be more important that the procedure to generate the DevNonce doesn't produce a value used in the past rather than to guarantee the randomness of the values. The randomness of DevNonce doesn't seem to be relevant for the security of the standard, especially because this value is not encrypted. Indeed, even if it is used as a "data" parameter to generate the two session keys, it is important that the value is different for every generation of these keys but not that it is random. So a pseudo-random number generator that doesn't produce a previously generated value or, easier, a sequential number could be adopted instead of a random number. Furthermore, adopting a sequential number, it is not required to increment relevantly the end-device memory, because it is necessary to store only the last used value.

However we have to consider that an end device can loose some network parameters and also the last used value of DevNonce, that we call LDN (Last DevNonce). If the server can't communicate to the end device the LDN, the end device is not able to join the network anymore. A solution provide that a default value of DevNonce, for example the value 0, to send at the server and communicate the loss of LDN. If the server received this default value, it understands that end device has lost the LDN. Then it sends the value to the end device, also not encrypted or encrypted with the AppKey. Since we adopted a default value of DevNonce to communicate the loss of LDN, the end device can send the value using the join request message.

Furthermore we have to consider that the end device can be unaware to have lost the LDN or that, due to an internal error, the value written in the register containing the LDN is different from the true LDN. If this happens, sending a join request message, the end device doesn't receive any answer by the server, because the value of LDN is wrong. After k join request without any answer, the end device can decide to set the value of DevNonce

to its default value and sends a join request. However, by the serever side, it may be better to introduce in the protocol a new type of message, different from join accept message, used by the server only to reply to a join request with default value. In this manner the server is not obliged to send the other network parameters contained in the join accept message every time it receives a request for the LDN.

It is important to stress the fact that, when this procedure is performed, the value of DevNonce is not reset. Indeed if the value of DevNonce is reset a malicious node M that records a join request with default value and it sends the recorded message to the server, it is able to reset the value of DevNonce at the server side but not at the side of the end device, because the node is unaware of the request and, if it is implementing the class A, the network server answer probably arrives in a time window different from the reception slots, causing dysfunction of the network.

In spite of these considerations, using this procedure, another problem is still present that makes the server vulnerable to a DoS attack. Indeed if a malicious node M records the join request with default value, it forces the server to respond at every request. If node M is able to send the request with high frequency it can overload the server that must be answer at every request. Furthermore the solution adopted to prevent replay attacks using previously recorded join accept message, is useless in this case since the end-device doesn't know the last used DevNonce.

In conclusion, using a sequential number instead of a random number, we prevent any possible problem related with invalid join request but we also introduce other issues that must be solved.

Another possible solution to adopt consists on increasing the size of DevNonce to 24 or 32 bits (hypothesizing that the minimum increment is of one octet). Considering the network server is implementing the policy of dropping the invalid join request, and the size of DevNonce is 24 (32) bits the probability in (3.2) is reduced of a factor 2^8 (2^{16}). In general, considering also the possibility of incrementing the DevNonce size as we require, the probability in (3.2) is halfed at every added bit. This solution can be useful if the estimated value of the valid join procedure performed by a node in its life is higher than N_D^{max}, so the probability in 3.7 is higher.

This solution is costly since we have to increment the size of join request message. However observing that, at the state of the art, the size of join request message is 18 bytes and the size of join accept message is 28 bytes, changing the size of DevNonce from 16 to 24 bits should not be a problem.

Apart these problems, another possible DoS attack is possible if we use the RegRssiWideband register as a random number generator, as recommended in [6]. In the next chapters we firstly explain how a receiver works,

focusing on how the RSSI (Received Signal Srength Indicator) is evaluated. Successively we analyze the recommended procedure, for a LoRa end device, for the generation of random numbers.

Chapter 4

Radio receiver architecture

In the second part of this work, we analyze the recommended procedure for the generation of a N-bit random number [6, Ch. 4], that is used to generate for example the DevNonce. In order to clearly understand the procedure we need to explain the architecture of the SX1272 receiver, focusing the attention on the thermal noise introduced by the device and by the environment.

4.1 General theory

The basic function of a radio receiver is distinguish signals from noise. The concept of noise covers both human-made and natural radio frequency signals. Human-made signals include all signals in the pass band other than the one being sought. In communications systems, the signal is some form of modulated (AM, FM, PM, OOK, etc.) periodic sine wave propagating as an electromagnetic (i.e., radio) wave [14].

A basic form of noise seen in systems is thermal noise. Even if the amplifiers in the receiver add no additional noise (they will), thermal noise will be found at the input due to the input resistance. If you replace the antenna with a resistor matched to the system impedance and totally shielded, noise still will be present. The noise is produced by the random motion of electrons inside the resistor. At all temperatures above absolute zero (about -273.16°C), the electrons in the resistor material are in random motion. At any given instant, a huge number of electrons will be in motion in all directions. The reason why there is no discernible current flow in one direction is that the motions cancel out each other, even over short time periods. The noise power present in a resistor is

$$P_w = kTBR \tag{4.1}$$

where $k = 1.38 \times 10^{-23}\ J/K$ is the Boltzmann constant, T is the temperature in Kelvin, R is the resistance in Ohm(Ω) and B is the bandwidth in Hz.

The SX1272 uses a superheterodyne receiver (Fig. 4.1). The purpose of a this receiver is to convert the incoming RF frequency to a single frequency where most of the signal processing takes place. The frontend section of the receiver consists of the radio frequency amplifier and any RF tuning circuits that may be used (A, B, and C in Fig. 4.1). In some cases, the RF tuning is very narrow and basically tunes one frequency. In other cases, the RF front-end tuning is broadband. In that case, bandpass filters are used. The translator consists of a frequency mixer(D) and a local oscillator(E). This section does the heterodyning. The output of the frequency translator is called the intermediate frequency.

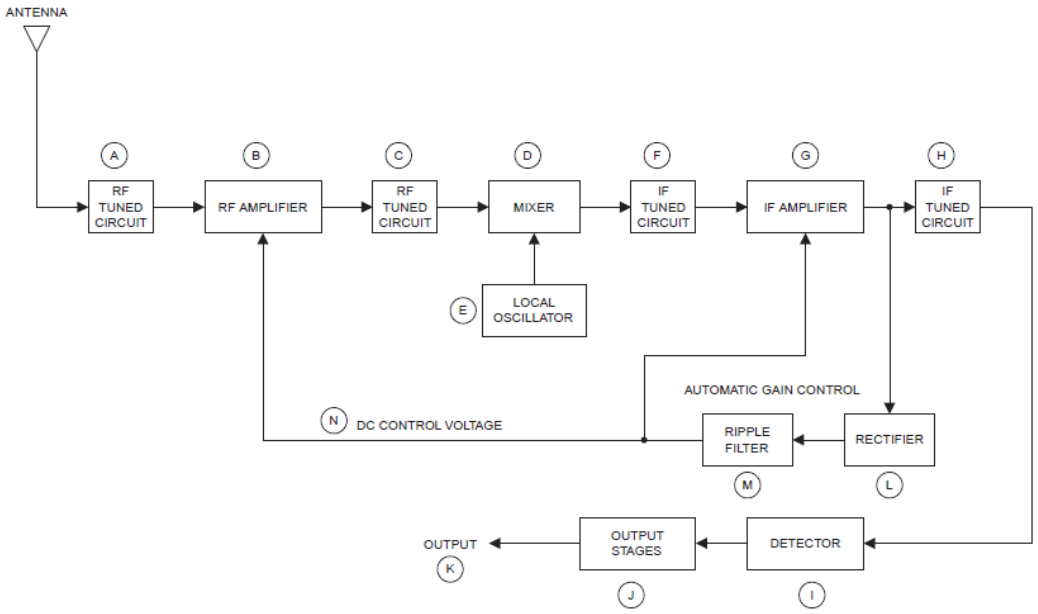

Figure 4.1: Radio receiver architecture [14].

The translator stage is followed by the intermediate frequency amplifier. The IF amplifier (F, G, and H) is basically a radio frequency amplifier tuned to a single frequency. The IF can be higher or lower than the RF frequency, but it always will be a single frequency. A sample of the IF amplifier output signal is applied to an automatic gain control (AGC) section (L and M). The purpose of this section is to keep the signal level in the output more or less constant. The AGC circuit consists of a rectifier and a ripple filter that produce a DC control voltage. The DC control voltage is proportional to the input RF signal level (N). It is applied to the IF and RF amplifiers to raise or lower the gain according to signal level. If the signal is weak, then the

gain is forced higher; and if the signal is strong, the gain is lowered. The result is to smooth out variations of the output signal level. The detector stage (I) is used to recover any modulation on the input RF signal. The type of detector depends on the type of modulation used for the incoming signal. Keyed CW signals will use a product detector. The output stages (J and K) are used to amplify and deliver the recovered modulation to the user. If the receiver is for broadcast use, then the output stages are audio amplifiers and loudspeakers.

In a datasheet of a receiver, usually, the input signal voltage (or power) is reported. Two forms of signal voltage are used for input voltage specification: source voltage (V_{EMF}) and potential difference (V_{PD}). The source voltage is the open terminal (with no load) voltage of the signal generator or source, while the potential difference is the voltage appears across the receiver antenna terminals with the load connected (see Fig. 4.2). The relation between V_{EMF} and V_{PD} is

$$V_{PD} = V_{EMF}\frac{R_{in}}{R_{in} + R_s} \tag{4.2}$$

where R_{in} is the receiver antenna input resistance and R_s is the source resistance. In matching condition ($R_{in} = R_s$) $V_{PD} = V_{EMF}/2$.

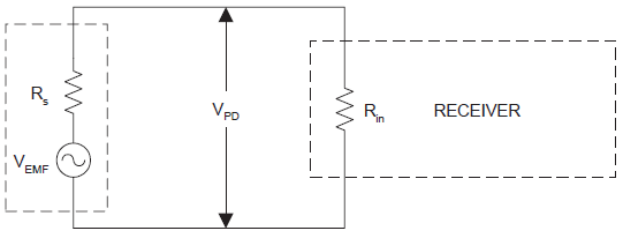

Figure 4.2: Receiver input voltage [14].

When the power input level is reported, instead of input signal voltage, often it is written in dBm unit. This unit refers to decibels relative to one milliwatt dissipated in a 50Ω resistive impedance.

Resuming the discussion about noise present in a receiver, usually it comes in a number of different guises, but for sake of this discussion, we divide them into two classes: sources external to the receiver and sources internal to the receiver. One can do little about the external noise sources, for they consist of natural and human-made electromagnetic signals that fall within the passband of the receiver. Fig. 4.3 shows an approximation of the external noise situation seen by receivers at different frequencies. One must select a

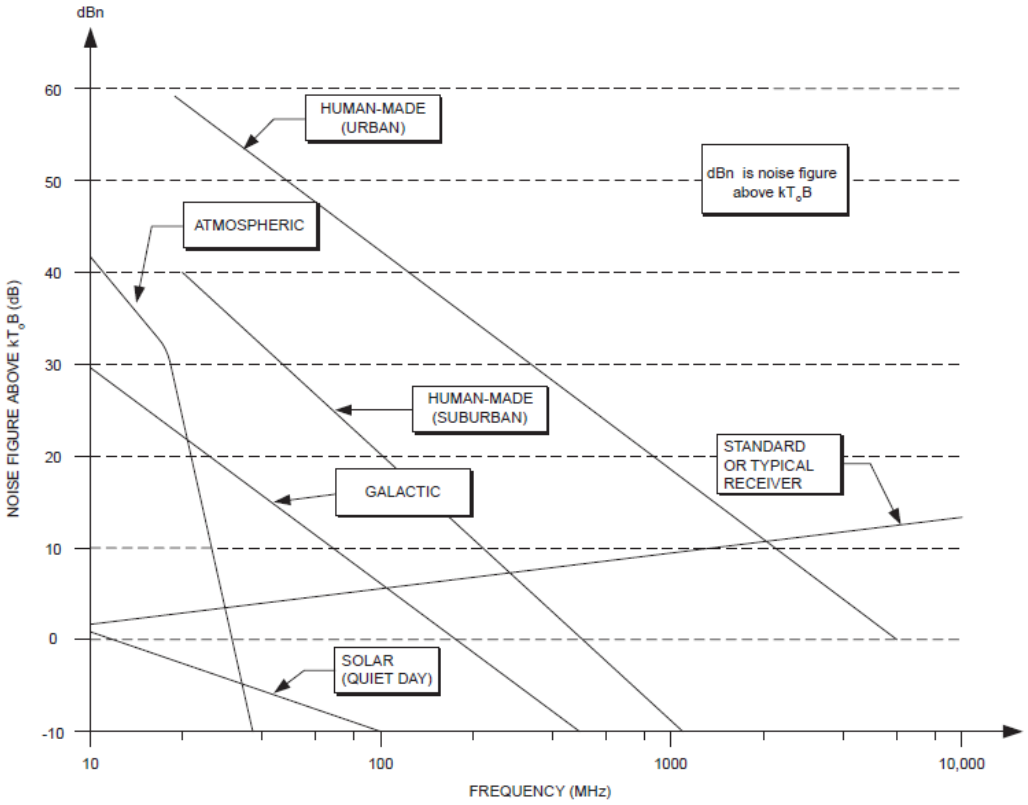

Figure 4.3: Noise sources throughout the bands [14].

receiver that can cope with external noise sources, especially if the noise sources are strong. Some natural external noise sources are extraterrestrial.

The receiver's internal noise sources are determined by the design of the receiver. Ideal receivers produce no noise of their own, so the output signal from the ideal receiver would contain only the noise present at the input along with the radio signal. But real receiver circuits produce a certain level of internal noise of their own. Even a simple fixed-value resistor is noisy. At any temperature above absolute zero (0K), electrons in any material are in constant random motion. Because of the inherent randomness of that motion, however, there is no detectable current in any one direction. In other words, electron drift in any single direction is cancelled over even short time periods by equal drift in the opposite direction. Electron motions therefore are statistically decorrelated. However, a continuous series of random current pulses is generated in the material, and those pulses are seen by the outside world as noise signals. If a perfectly shielded 50 Ω resistor is connected across the antenna input terminals of a radio receiver, the noise level at the receiver output will increase by a predictable amount over the short-circuit noise level.

Noise signals of this type are called by several names: thermal agitation noise, thermal noise, or Johnson noise. This type of noise also is called white noise, because it has a very broadband (nearly Gaussian) spectral density. The thermal noise spectrum is dominated by mid-frequencies (104–105 Hz) and essentially is flat. The term white noise is a metaphor developed from white light, which is composed of all visible color frequencies. The expression for the power pf this noise is

$$P_w = \frac{V_w^2}{R} = kTB \tag{4.3}$$

where P_w is the noise power in watt(W).

The noise performance of a receiver or amplifier can be defined in three different but related ways: noise factor, noise figure, and equivalent noise temperature.

For components such as resistors, the noise factor (NF) is the ratio of the noise produced by a real resistor to the simple thermal noise of an ideal resistor. The noise factor of a radio receiver (or any system) is the ratio of output noise power ($P_{w,out}$) to input noise power ($P_{w,in}$):

$$NF = \frac{P_{w,out}}{P_{w,in}}. \tag{4.4}$$

To make comparison easier, the noise factor is usually measured at the standard temperature of $290K$. The noise figure, instead, is the noise factor converted to decibel notation

$$F = 10 \log_{10}(NF) \tag{4.5}$$

Finally the equivalent noise temperature is a means for specifying noise in terms of an equivalent noise temperature; that is the noise level that would be produced by a matching resistor (e.g. 50Ω) at that temperature (expressed in degrees Kelvin). Note that the equivalent noise temperature, T_e, is not the physical temperature of the amplifier but rather a theoretical construct that is an equivalent temperature producing that amount of noise power in a resistor. The noise temperature is related to the noise factor by

$$T_e = (NF - 1)T_0. \tag{4.6}$$

This analysis about noise doesn't consider that thermal noise is a random process where every sample is a Gaussian random variable with zero mean and variance

$$\sigma_w^2 = kTRB \tag{4.7}$$

So at the receiver, at any time t, there is a noise voltage sample of value $w(t)$ with probability density function

$$p_{w(t)}(x) = \frac{1}{\sqrt{2\pi\sigma_w^2}} e^{-\frac{x^2}{2\sigma_w^2}}. \tag{4.8}$$

and with voltage noise power

$$P_{w(t)} = \frac{w(t)^2}{R}. \tag{4.9}$$

and in average the power of noise is given by (4.3).

4.2 SX1272 Receiver model

As we can see in the next chapter, we are interested on the mechanisms and procedures that permit to evaluate the RSSI, because this value is exploited for the generation of random numbers. In order to evaluate the theoretical value written in the RSSI register, considering also the presence of noise, we used the model of Fig. 4.4 as the receiver. In this model some assumptions have been done:

- The antenna gain is 0dB.
- The electrical power is one fourth the electromagnetic power for the matching condition.
- The receiver resistance is 50Ω.
- The value of maximum RF input power level reported in SX1272 datasheet corresponds with the power of received electromagnetic wave.
- The electrical voltage assumes non negative values.
- The received signal has constant amplitude in time or it can be modeled as a Gaussian random variable.
- The noise is due only to thermal noise.
- The noise temperature is due to temperature of environment and noise figure of LNA.
- The gain of the receiver is due to the gain of LNA.
- The RSSI register is before the channel filters of the receiver.

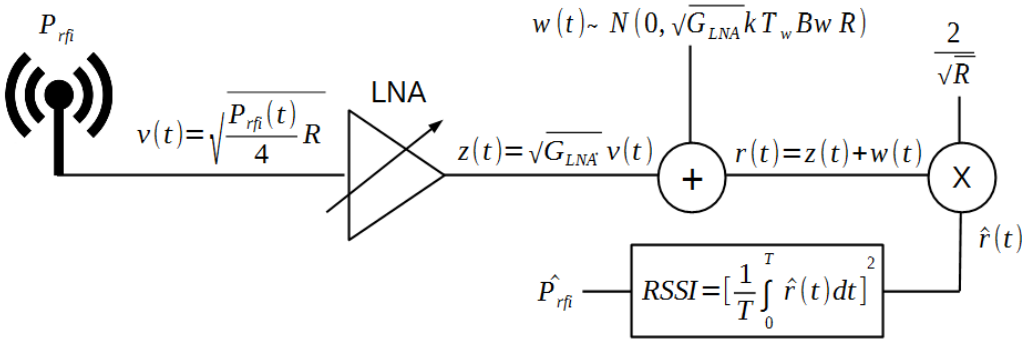

Figure 4.4: SX1272 receiver model.

- Other components of the receiver (mixer, etc.) don't influence the RSSI value written in the register.

Let's explain the model in Fig. 4.4. First of all we have to clarify that, observing experimentally the values of RSSI written in the registers, it is probable that the gain introduced by LNA is not considered for the evaluation of RSSI, that is the voltage signal $r(t)$ is not divided by $\sqrt{G_{LNA}}$ and $\hat{P}_{rfi}$ is almost smaller than P_{rfi} by a factor G_{LNA}. However, even if this observation is false, for our purposes it is important to consider the attenuated signal $r(t)$ because probably and reasonably $r(t)$ is the analog signal that is later quantized. It is possible that also the factor due to the matching condition and receiver resistance $(2/\sqrt{R})$ is not considered and the RSSI value is simply evaluated through the square of $r(t)$. However in our model we supposed that the receiver consider the terms due to matching condition and resistance impedance.

Because of the matching condition (Fig. 4.2 with $R_s = R_{in}$) the electrical power due to received signal is

$$P_{el}(t) = P_{rx}(t)/4. \tag{4.10}$$

So the received power produces, in the device, an electrical voltage

$$v(t) = \sqrt{\frac{P_{el}(t)}{4}R} \tag{4.11}$$

that is then multiply by the square root of the gain of LNA. However every electrical device generates thermal noise. For the superposition principle we can consider the noise as an added voltage modeled as a Gaussian random variable with zero mean and variance given by (4.7) and pdf given by (4.8).

In this model, the highest noise contribution is given by LNA. The noise temperature T_w of the receiver chain is

$$\begin{aligned} T_w &= T_0 + T_{LNA} \\ T_0 &= 290K \\ T_{LNA} &= (F_{LNA} - 1)T_0 \end{aligned} \tag{4.12}$$

The LNA noise figure depends on the input signal power, because in order to increment the precision, the LNA uses different gains (and so different noise figures) based on the power input level. We report in Tab. 4.1 values of different gains and noise figures for different levels with the default range of input power for each level, where Ref is the reference level that can be set as the sensitivity level S. (see also Fig. 4.5).

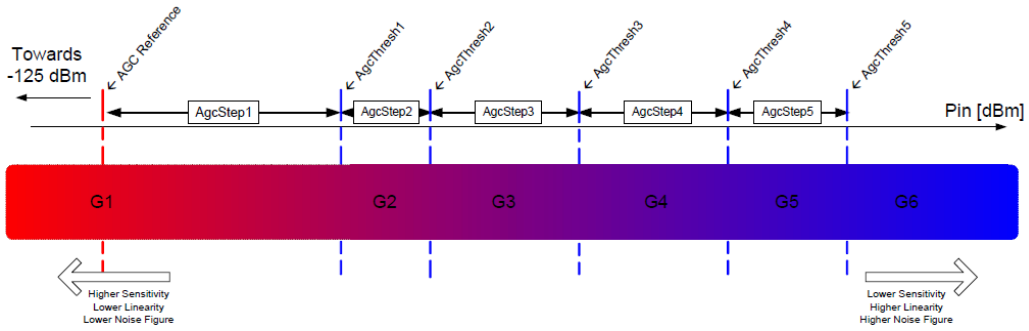

Figure 4.5: Different gain level of LNA [10].

Rx input Level $P_{in}[dBm]$	Gain[dB]	Noise Figure[dB]
$P_{in} \leq Ref + 14$	0	7
$Ref + 14 < P_{in} \leq Ref + 19$	-6	11
$Ref + 19 < P_{in} \leq Ref + 30$	-12	16
$Ref + 30 < P_{in} \leq Ref + 43$	-24	26
$Ref + 43 < P_{in} \leq Ref + 54$	-26	34
$P_{in} > Ref + 54$	-48	44

Table 4.1: Parameters of Low Noise Amplifier for different levels[10]

Usually the RSSI is not an instant value but it is evaluated averaging the instant value of voltage in a time interval T, having a new value of RSSI every T seconds. In the LoRa networks $T = 1/B$, where B is the bandwidth of the transmitted or received signal. For example, choosing $B = 125$ kHz we have $T = 8\ \mu s$. Since the voltage signal is sampled, the RSSI is given averaging the values of K samples. Considering the factor $4/R$ due to matching condition and resistance impedance

$$RSSI = \frac{1}{K}\sum_{n=1}^{K}\frac{4}{R}\left(v(n)+w(n)\right)^2 = \frac{1}{K}\sum_{n=1}^{K} z(n)^2 \tag{4.13}$$

where $v(n)$ and $w(n)$ are the sampled voltages due to signal and noise respectively. Hypothesizing that the voltage due to the received signal is constant in time, afterwards we will write v instead of $v(n)$. Since $z(n)$ is a Gaussian random variable $\mathcal{N} \sim \left(\frac{2v}{\sqrt{R}}, \frac{4\sigma_w^2}{R}\right)$, if we divide (4.13) by the variance of $z(n)$ and multiply by K, we obtain a noncentral chi squared random variable $NC\chi^2$, that is

$$RSSI = \frac{1}{K}\sum_{n=1}^{K} z(n)^2 = \frac{4\sigma_w^2}{KR}\sum_{n=1}^{K}\left(\frac{v+w(n)}{\sigma_w}\right)^2 = \frac{4\sigma_w^2}{KR} NC\chi^2. \tag{4.14}$$

In our case $NC\chi^2$ has K degrees of freedom and

$$\begin{aligned} \lambda &= \sum_{n=1}^{K} E\left[z(n)\right]^2 = K\left(\frac{v}{\sigma_w}\right)^2 \\ E[NC\chi^2] &= K + \lambda = K\left(1+\left(\frac{v}{\sigma_w}\right)^2\right) \\ var[NC\chi^2] &= 2K\left(1+2\left(\frac{v}{\sigma_w}\right)^2\right) \end{aligned} \tag{4.15}$$

where λ is called noncentrality parameter. Then the average and variance of RSSI are

$$\begin{aligned} E[RSSI] &= \frac{4\sigma_w^2}{KR} E[NC\chi^2] = \frac{4}{R}\left(\sigma_w^2 + v^2\right) \\ var[RSSI] &= \left(\frac{4\sigma_w^2}{KR}\right)^2 var[NC\chi^2] = \frac{32\sigma_w^2}{KR^2}(\sigma_w^2 + 2v^2). \end{aligned} \tag{4.16}$$

We can observe that the value of $var[RSSI]$ decreases increasing K. So the higher is the number of samples the lower is the uncertainty on the value of RSSI. So the maximum uncertainty is obtained with $K = 1$.

Furthermore the variance depends also on the square of the signal and noise voltage, that is it depends on the power of the signal and noise; in particular the variance is directly proportional to the power of the received signal.

However the hypothesis that the received signal is constant can be too tight. If the signal is not constant, it can be modeled as a Gaussian $v \sim \mathcal{N}(m_v, \sigma_v^2)$. In this case the equation of mean and variance in (4.16) are the same with $\sigma_w^2 + \sigma_v^2$ instead of σ_w^2 and m_v instead of v, i.e.

$$\begin{aligned} E[RSSI] &= \frac{4}{R}\left(\sigma_w^2 + \sigma_v^2 + m_v^2\right) \\ var[RSSI] &= \frac{32(\sigma_w^2 + \sigma_v^2)}{KR^2}(\sigma_w^2 + \sigma_v^2 + 2m_v^2). \end{aligned} \tag{4.17}$$

Another hypothesis is that the device averages the voltage samples and then compute the square of the mean voltage instead of averaging the square of the voltage samples. In this case

$$\begin{aligned} RSSI &= \frac{4}{R}\left(\frac{1}{K}\sum_{n=1}^{K} v(n) + w(n)\right)^2 = \frac{4}{R}\left(\frac{1}{K}\sum_{n=1}^{K} z(n)\right)^2 = \frac{4}{R}\left(\hat{z(n)}\right)^2 = \\ &= \frac{4\sigma_z^2}{R} NC\chi^2 \end{aligned} \tag{4.18}$$

where, in general, $z \sim (m_z, \sigma_z^2)$, $\hat{z} \sim \left(m_z, \frac{1}{K}\sigma_z^2\right)$ and the RSSI is a noncentral chi-squared with degree of freedom equal to 1, but with a variance that is decreasing with the increment of K. So in all the cases, when there is a received signal, the value of RSSI is a noncentral chi-squared.

Instead, when a received signal is not present, the value of RSSI depends only on the noise and the RSSI isn't a noncentral chi-squared because the noise has zero mean and the noncentral chi-squared required $\lambda > 0$. However a noncentral chi-squared with $\lambda = 0$ corresponds with the chi-squared distribution, i.e.

$$RSSI = \frac{4}{KR}\sum_{n=1}^{K} w(n)^2 = \frac{4\sigma_w^2}{KR}\sum_{n=1}^{K}\frac{w(n)^2}{\sigma_w^2} = \frac{4\sigma_w^2}{KR}\chi^2. \tag{4.19}$$

In this case mean and variance are

$$\begin{aligned} E[RSSI] &= \frac{4\sigma_w^2}{R} \\ var[RSSI] &= \frac{32\sigma_w^4}{KR^2}. \end{aligned} \tag{4.20}$$

Also in this case, if the average is evaluated on the voltage sample, instead of squared voltage sample, $K = 1$.

However in the register we don't write analog values of RSSI, but a quantized version that uses a logarithmic scale (dBm) with step of 0.5 dB. Then the probability that the value R is written in the register is

$$\begin{aligned} Pr\left[RSSI = R\right] =& Pr[T_1 < RSSI < T_2] = \\ &= \frac{4\sigma_z^2}{KR} \int_{T_1}^{T_2} pdf(NC\chi^2)dx \end{aligned} \tag{4.21}$$

with $T_1 = 10^{(R-0.25-30)/10}$ and $T_2 = 10^{(R+0.25-30)/10}$ the two thresholds of the level R of the quantization in linear scale and

$$pdf(NC\chi^2) = \frac{1}{2} e^{-(x+\lambda)/2} \left(\frac{x}{\lambda}\right)^{k/4-1/2} I_{k/2-1}(\sqrt{\lambda x}) \tag{4.22}$$

where $I_\nu(y)$ is a modified Bessel function of the first kind.

Unfortunately the SX1272 transceiver datasheet [10] doesn't give any information about the number of samples that is averaged and so it is difficult to predict the behavior of the transceiver theoretically. However in Figg. 4.6 and 4.7 we plot several chi-squared and noncentral chi-squared distribution. In particular in Fig. 4.6 we plot the chi-squared distribution (hypothesizing that the value of RSSI is given only by Gaussian noise) for different values of K. The values set for K are multiple of 2, because in [10] for the RegRssiValue is reported that the number of samples for the evaluation of RSSI is multiple of 2. Then is reasonable that also for RegRssiWideband (the register that we are interested on for the generation of random numbers) the number of samples is a multiple of 2. The value of RSSI in the x-axis is given in dBm hypothesizing that the noise is thermal with $T_0 = 290K$ and its variance is given by (4.7), and with the hypothesis that the average is due on power sample. However if these hypotheses are wrong the distribution doesn't change, but it change the corresponding values of RSSI. In Fig. 4.7, instead, we plot the noncentral chi-squared for different values of $\frac{v}{\sigma_w^2}$ with $K = 64$, hypothesizing to have a constant voltage v given by received signal and thermal noise with variance σ_w. However the results are the same even in the case the received signal can be modeled with Gaussian pdf, substituting v with the mean of the total voltage signal and σ_w^2 with the total variance. The value of $K = 64$ has been choice observing the similarity between the quantized RSSI pmf with only thermal noise and the experimental pmf of RSSI, reported in the next chapter, when the device is inside a metal box. However using a different value of K we observed a similar behavior (see Fig. 4.8)

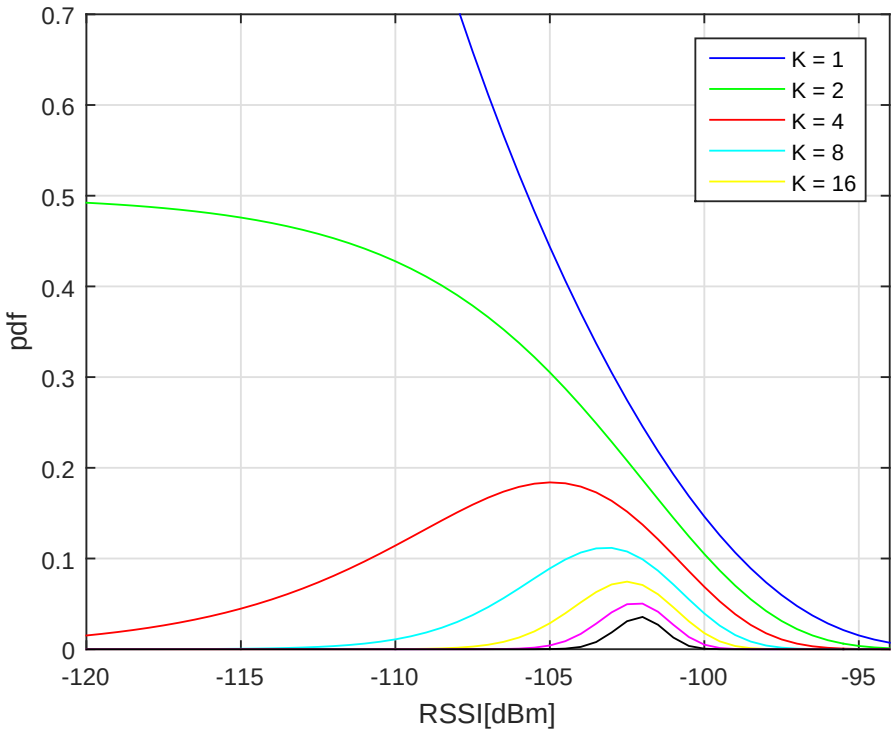

Figure 4.6: Chi-squared distribution for different values of K.

However the trend of pdf of chi-squared and noncentral chi-squared can be misleading if we compare them with the experimental pmf of RSSI values. Indeed the value of RSSI is quantized: so we don't have a pdf but a pmf and the probability that the RSSI is equal to R is given in (4.21). In Figg. 4.9 and 4.10 we plot the theoretical pmf of quantized chi-squared and noncentral chi-squared distribution with the parameters of pdf. Moreover in Fig. 4.11 we plot the quantized noncentral chi-squared with $K = 8$ in order to compare the trends with that with $K = 64$. We can observe that for both distributions we have only one peak and the higher is K the more narrow is the peak. Furthermore for the quantized noncentral chi-squared also if the ratio $\frac{v}{\sigma_w}$ is higher, the peak is more narrow.

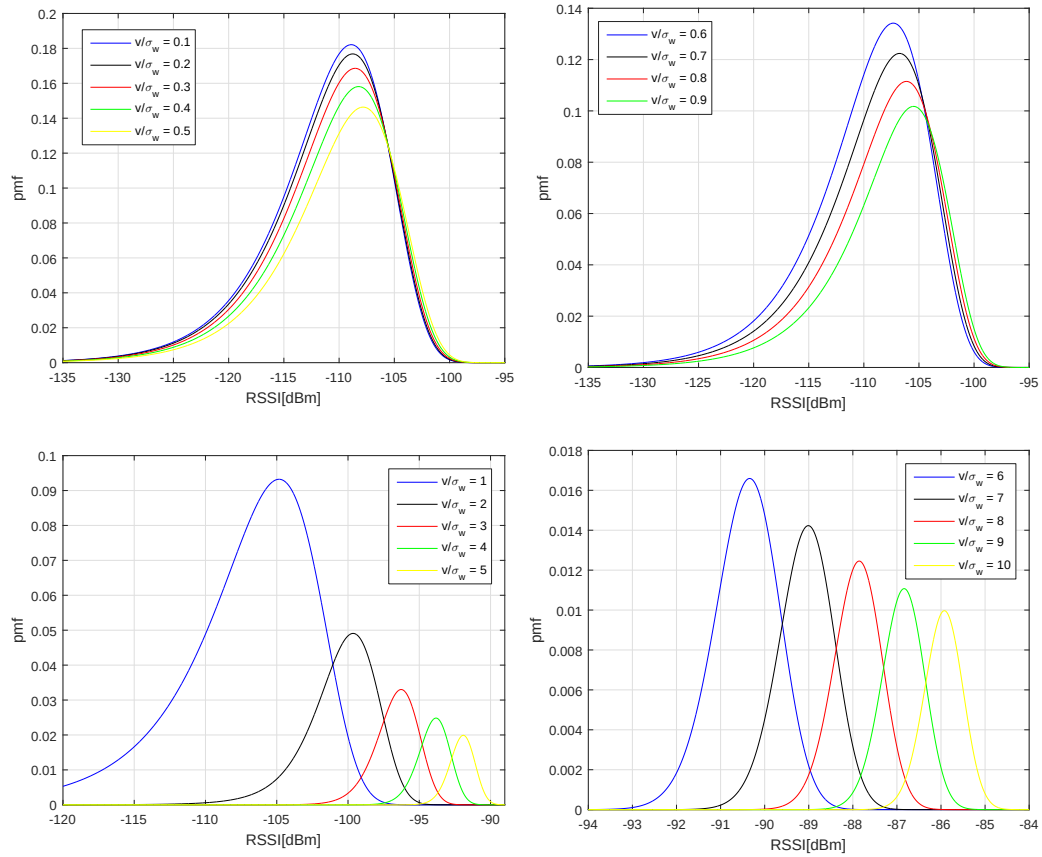

Figure 4.7: Noncentral chi-squared distribution with $K = 64$ for different values of $\frac{v}{\sigma_w}$.

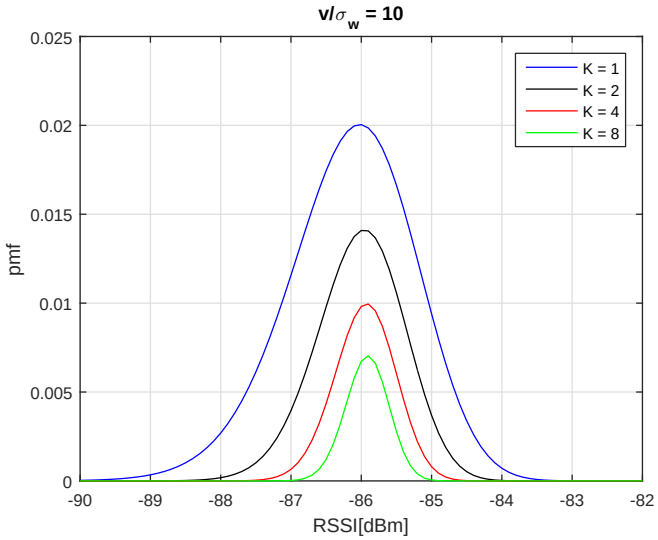

Figure 4.8: Noncentral chi-squared distribution with different values of K and $\frac{v}{\sigma_w} = 10$.

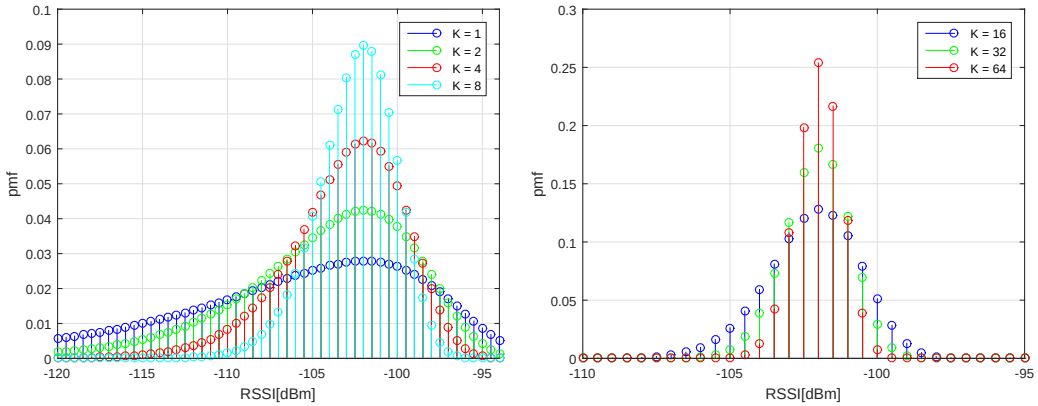

Figure 4.9: Quantized chi-squared distribution for different values of K.

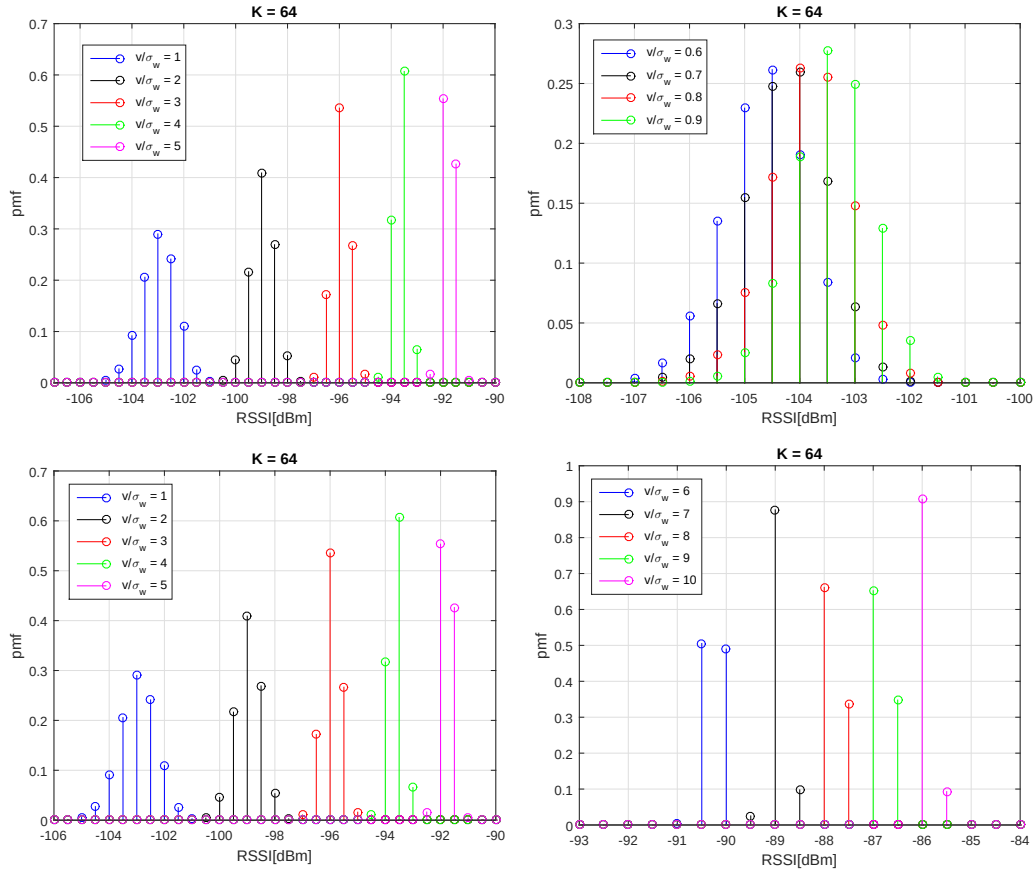

Figure 4.10: Quantized noncentral chi-squared distribution with $K = 64$ for different values of $\frac{v}{\sigma_w^2}$.

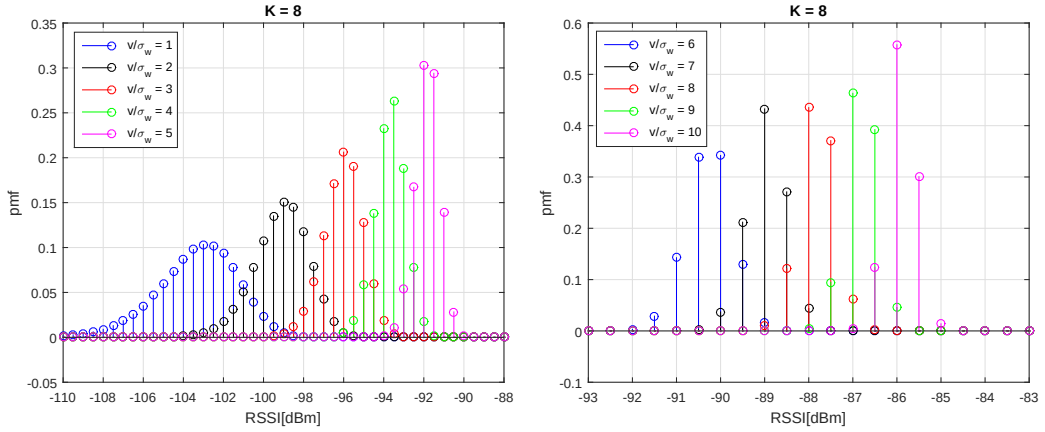

Figure 4.11: Quantized noncentral chi-squared distribution with $K = 8$.

Chapter 5

Random number generation

A random-number generator (RNG) is a computational or physical device designed to generate a sequence of numbers or symbols that cannot be reasonably predicted better than by a random chance. Random number generators have applications in gambling, statistical sampling, computer simulation, cryptography, completely randomized design, and other areas where producing an unpredictable result is desirable. Generally, in applications having unpredictability as the paramount, such as in security applications, hardware generators are generally preferred over pseudo-random algorithms, where feasible [17].

Another important application that requires RNG is the software cryptography such as SSH, IPSEC, TLS, S/MIME, PGP, DNSSEC, and Kerberos. These systems provide substantial protection against snooping and spoofing. However at the heart of all cryptographic systems is the generation of secret, unguessable (i.e. random) numbers. The lack of generally available facilities for generating such random numbers (that is, the lack of general availability of truly unpredictable sources) forms an open wound in the design of cryptographic software. For the software developer who wants to build a key or password generation procedure that runs on a wide range of hardware, this is a very real problem. Note that the requirement is for data that an adversary has a very low probability of guessing or determining. This can easily fail if pseudo-random data is used that meets only traditional statistical tests for randomness, or that is based on limited-range sources such as clocks. Sometimes such pseudo-random quantities can be guessed by an adversary searching through an embarrassingly small space of possibilities [16].

Concerning LoRaWAN protocol randomness or pseudo-randomness is required for:

- selection of the channel for every transmission;

- implementation of ALOHA-type protocol;
- Slot randomization in case of Class B end-device;
- Generation of DevNonce and AppNonce.
- Generation of AppKey.

In particular only DevNonce must be a "true" random number that must be generated from the end-device, while the other quantities are pseudorandom and/or are generated by other entities, such as network server or application provider. However for the generation of DevNonce, a LoRa end-device requires a Random Number Generator.

In 2005 [RFC 4086][16] pointed out many pitfalls in using poor entropy sources or traditional pseudo-random number generation techniques for generating such quantities. It recommends the use of truly random hardware techniques and shows that the existing hardware on many systems can be used for this purpose. It provides suggestions to ameliorate the problem when a hardware solution is not available, and it gives examples of how large such quantities need to be for some applications.

Generally speaking two different types of random quantities may be wanted. In the case of human-usable passwords, the only important characteristic is that they be unguessable. It is not important that they may be composed of ASCII characters, so the top bit of every byte is zero, for example. On the other hand, for fixed length keys and the like, one normally wants quantities that appear to be truly random, that is, quantities whose bits will pass statistical randomness tests.

5.1 Theory

Usually an adversary can try to determine the key (the random number) by trial and error. The probability of an adversary succeeding at this must be made acceptably low, depending on the particular application. The size of the space the adversary must search is related to the amount of key "information" present, in an information-theoretic sense. This depends on the number of different secret values possible and the probability of each value, as follows

$$H(X) = E[-\log_2(P(X))] = \sum_{i=1}^{n} -P(x_i)\log_2\left(P(x_i)\right) \tag{5.1}$$

where $H(X)$ is the entropy of a discrete random variable X with possible values $\{x_1, \ldots, x_n\}$ and probability mass function P(X). If there are 2^n different

values of equal probability, then n bits of information are present and an adversary would have to try, on the average, half of the values, or 2^{n-1}, before guessing the secret quantity. If the probability of different values is unequal, then there is less information present, and fewer guesses will, on average, be required by an adversary. In particular, any values that an adversary can know to be impossible or of low probability can be initially ignored by the adversary, who will search through the more probable values first. Moreover if for example, we consider a cryptographic system that uses 128-bits keys derived using a fixed pseudo-random number generator that is seeded with an 8-bits seed, then an adversary needs to search through only 256 keys (by running the pseudo-random number generator with every possible seed), not 2^{128} keys as may at first appear to be the case. Only 8 bits of information are in these 128-bits keys.

While the above analysis is correct on average, it can be misleading in some cases for cryptographic analysis where what is really important is the work factor for an adversary. For example, assume that there is a pseudo-random number generator generating 128-bits keys, as in the previous paragraph, but that it generates zero half of the time and a random selection from the remaining $2^{128} - 1$ values the rest of the time. The Shannon equation above says that there are 64 bits of information in one of these key values, but an adversary, simply by trying the value zero, can break the security of half of the uses, albeit a random half. Thus, for cryptographic purposes, it is also useful to look at other measures, such as min-entropy, defined as

$$\text{min-entropy} = -\log_2 \left(\max_i P(x_i) \right). \tag{5.2}$$

We can observe that we get 1 bit of min-entropy for our new hypothetical distribution, as opposed to 64 bits of classical Shannon entropy.

Statistically tested randomness in the traditional sense is not the same as the unpredictability required for security use. For example, the use of a widely available constant sequence, such as the random table from the CRC Standard Mathematical Tables, is very weak against an adversary. An adversary who learns of or guesses it can easily break all security, future and past, based on the sequence. On the other hand, taking successive rolls of a six-sided die and encoding the resulting values in ASCII would produce statistically poor output with a substantial unpredictable component. So note that passing or failing statistical tests doesn't reveal whether something is unpredictable or predictable.

The National Institute of Standards and Technology in [15] specifies the design principle and requirements for the entropy sources used by Random Bit Generators (RBGs), and the tests for the validation of entropy sources.

The development of entropy sources that provide unpredictable output is difficult, and providing guidance for their design and validation testing is even more so. An entropy source that conforms the Recommendation can be used by RBGs to produce sequence of random bits or can be used by pseudo-random bit generators as a seed value.

In order to build a cryptographic RBG we need:

- a source of random bits (the entropy source);
- an algorithm, typically a Deterministic Random Bit Generator (DRBG), that accumulates and provides the random numbers to the application;
- a way to combine the first two components for the application.

5.2 Entropy source

Concerning the entropy source, the developer must be able to accurately estimate the amount of entropy that can be provided by sampling the noise source, considering also the interaction of the entropy source with other components, and taking care if the output from the noise source is biased.

Entropy sources tend to be very implementation dependent. Once one has gathered sufficient entropy, it can be used as the seed to produce the required amount of cryptographically strong pseudo-randomness, after being de-skewed or mixed as necessary. Thermal noise (sometimes called Johnson noise in integrated circuits) or a radioactive decay source and a fast, free-running oscillator would do the trick directly. This is a trivial amount of hardware, and it could easily be included as a standard part of a computer system's architecture. Most audio (or video) input devices are usable. Furthermore, any system with a spinning disk or ring oscillator and a stable (crystal) time source or the like has an adequate source of randomness. All that's needed is the common perception among computer vendors that this small additional hardware and the software to access it is necessary and useful.

In [15] it is described an entropy source model, composed by a noise source, an optional conditioning component and a health testing component (See Fig. 5.1).

Let's now examine separately the three components of an entropy source model.

Noise source The noise source is the root for the entropy source and for the RBG. This is the component that contains the non-deterministic,

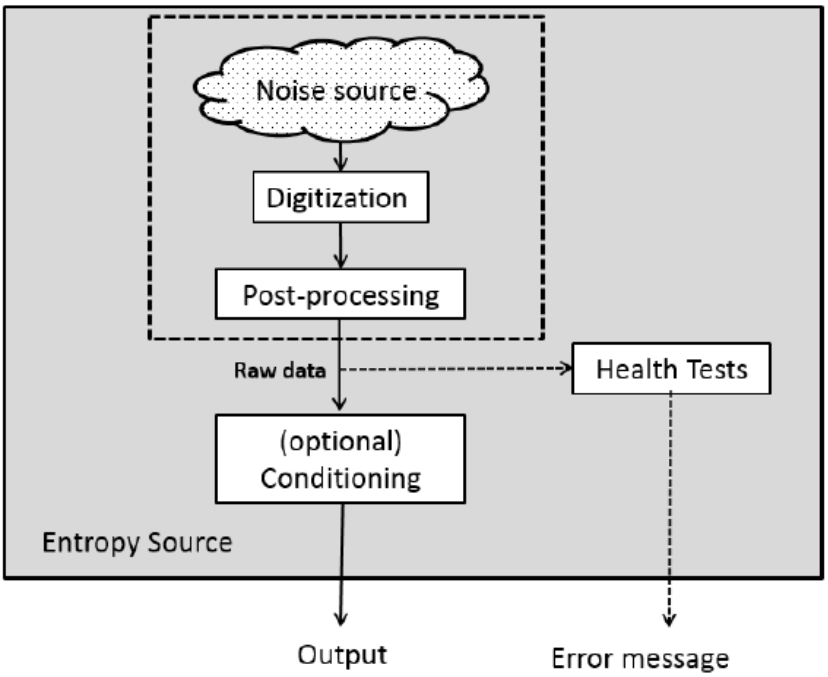

Figure 5.1: Entropy Source Model [15].

entropy-providing activity that is ultimately responsible for the uncertainty associated with the bitstrings output by the entropy source. The noise source must be sampled (if it is analog) and digitized (if it isn't produce binary data), Moreover some post-processing operations may be fulfilled in order to reduce statistical biases and increase the entropy rate of the resulting output, that is called *raw data*. If the noise source fails to generate random outputs, no other component in the RBG can compensate for the lack of entropy. In some situations multiple noise sources may be combined in order to increase the total entropy. If the noise sources are independent, their entropy assessments can be added, however the total entropy is harder to estimate.

Conditioning component The optional conditioning component is a deterministic function responsible for reducing bias and/or increasing the entropy rate of the resulting output bits.

Health tests Health tests are intended to ensure that the noise source and the entire entropy source continue to operate as expected. The end goal is to obtain assurance that failures of the entropy source are caught quickly and with a high probability. Another aspect of health testing strategy is determining likely failure modes for the entropy source and, in particular,

for the noise source. Health tests are expected to include tests that can detect these failure conditions.

5.3 Generation of random numbers with SX1272

The SX1272 transceiver features the LoRa™ long-range modem that provides ultra-long-range spread spectrum communication and high interference immunity while minimizing current consumption. At the state of the art every end-device joining a LoRa network uses this transceiver for the transmission and reception of the messages. A procedure to generate a random value must be implemented by the end-device in order to create random numbers required in the protocol, such as the DevNonce. Semtech Corporation [6, Ch.4] for the generation of an N bits random number recommends, when the end-device has the SX1272 transceiver, to perform N read operation of the least significant bit of the register RegRssiWideband (address 0x2c). Moreover in order to perform the read operations the following parameters must be set:

SX1272 Register (address)	Register bit field (bit #)	Values	Note
RegOpMode (0x01)	LongRangeMode[7]	'1'	LoRa mode enable
	Mode[2:0]	'101'	Receive Continuous mode
RegModemConfig1 (0x1D)	Bw[7:6]	'00'	'00' for 125kHz modulation Bandwidth
	CodingRate[5:3]	'001'	4/5 error coding rate
	ImplicitHeaderModeOn[2]	'0'	Packet have up-front header
	RxPayloadCrcOn[1]	'1'	CRC enable
	LowDataRateOptimize[0]	'0'	'0' when Spreading Factor is <= 10
RegModemConfig2 (0x1E)	SpreadingFactor[7:4]	'0111'	'0111' (SF7) = 6kbit/s
	AgcAutoOn[2]	'1'	
	SymbTimeout[1:0]	'00'	

RSSI means Received Signal Strength Indicator and it is a measurement of the power present in a received radio signal. In particular the value from RegRssiWideband is derived from a wideband (4 MHz) signal strength at the receiver input. In particular the LSB of this value constantly and randomly changes for the presence of the noise channel.

This procedure for the generation of DevNonce is also demonstrated in the source code of the end device implementing the LoRaWAN protocol, that is available online from GitHub[7], in particular the function for the generation of random value through RegRssiWideband register is present in the file SX1272.c and the employment of that function for the generation of DevNonce is documented in the file LoRaMAC.c. From the file we can observe that a bit is generated every 1 ms, that is, in the hypothesis that the entropy of the source is maximum, this procedure has a rate of 1000 bit/s.

Modeling the procedure as in Fig. 5.1, we have that the noise source is the RSSI of a wideband signal (4 MHz). This value contains a random component due to radio channel behavior (reflections, fading, shadowing, interference) and noise (in particular thermal noise) introduced by the receiver's components. The output of the digitization process corresponds to the value written in the RegRssiWideband register that goes from 0 to 127 dBm, with step of 0.5 dB. Finally the operation of considering only the least significant bit of the value written in the register corresponds with the post-processing process or with the conditional component, that has the function to reduce the biases of the entropy sources.

How we can immediately notice, in this procedure, health tests, ensuring the entire entropy source continues to operate as expected, are missing. In the following sections we analyze some conditions that can bring the entropy source to not operate correctly.

Another critical aspect is the inadequate documentation about the procedure. In [15] the following requirements are necessary:

≪ ***Requirements on the Entropy Source***

1. *The entire design of the entropy source shall be documented, including the interaction of the components[...]. The documentation shall justify why the entropy source can be relied upon to produce bits with entropy.*

2. *Documentation shall describe the operation of the entropy source, including how the entropy source works, and how to obtain data from within the entropy source for validation testing.*

3. *Documentation shall describe the range of operating conditions under which the entropy source is claimed to operate correctly (e.g., temperature range, voltages, system activity, etc.). Analysis of the entropy source's behavior at the edges of these conditions shall be documented, along with likely failure modes.*

4. *The entropy source shall have a well-defined (conceptual) security boundary[...]. This security boundary shall be documented; the documentation shall include a description of the content of the security boundary.[...]*

Requirementes on the Noise Source

1. *The operation of the noise source shall be documented; this documentation shall include a description of how the noise source works and rationale about why the noise source provides acceptable entropy output, and should reference relevant, existing research and literature. Documentation shall also include why it is believed that the entropy rate does not change significantly during normal operation.*

2. *Documentation shall provide an explicit statement of the expected entropy rate and provide a technical argument for why the noise source can support that entropy rate. This can be in broad terms of where the unpredictability comes from and a rough description of the behavior of the noise source (to show that it is reasonable to assume that the behavior is stable).*

3. *The noise source state shall be protected from adversarial knowledge or influence to the greatest extent possible. The methods used for this shall be documented, including a description of the (conceptual) security boundary's role in protecting the noise source from adversarial observation or influence.*

4. *Although the noise source is not required to produce unbiased and independent outputs, it shall exhibit random behavior; i.e., the output shall not be definable by any known algorithmic rule. Documentation shall indicate whether the noise source produces IID data or non-IID data. This claim will be used in determining the test path followed during validation. If the submitter makes an IID claim, documentation shall include rationale for the claim.*

5. *The noise source shall generate fixed-length bitstrings. A description of the output space of the noise source shall be provided. Documentation shall specify the fixed sample size (in bits) and the list (or range) of all possible outputs from each noise source.* ≫

None of these requirements (or at least few of them) are satisfied by the procedure for generation of random numbers recommended by Semtech for SX1272. In particular:

1. the design of the entropy source is not documented;

2. the range of operating conditions under which the entropy source is claimed to operate correctly is not reported, especially the bandwidth of the signal for which the value of RSSI is derived;

3. the security boundary are not defined;

4. documentation about entropy output and entropy rate is not reported;

5. methods for protecting the noise source from adversarial knowledge or influence are not specified;

6. it is not specified if the procedure produces IID or non-IID data;

7. there isn't documentation about implemented tests;
8. there isn't any citation to reports, articles or other literature about the procedure;
9. a mechanism to verify continuously the correctness of the random number generation is not implemented;
10. there isn't explanation about why the receiver must be set with the recommended parameters;

The insufficient documentation about the procedure for the generation of random number should bring any LoRa device's owner, equipped with SX1272 transceiver, to not used the recommended procedure. Anyway the scarce documentation is not a proof that the procedure doesn't work.

5.4 Hacking the SX1272 RNG

Theoretically speaking the randomness of the number generated through the procedure described above could be not achieved if one of the following situations happens:

a) The received power is so high that the receiver saturates;

b) The receiver doesn't saturate but the value of RSSI doesn't depend on noise (or other random phenomena).

In case a) the RegRssiWideband register is storing the maximum value (all ones in binary format) since the power present in the received radio signal is higher than maximum value and its value remains constant in time. Indeed, if saturation value is significantly exceeded, the probability that noise brings the power below saturation is very low. In case b), since the RegRssiWideband value is quantized, it is possible that in some circumstances, for example with high and constant received power, the value written in the register may change negligibly with high probability due to the noise, becoming de facto constant in time (if the received power is constant).

In these two cases the LoRa end-device is susceptible to a Denial of Service (DoS) attack. Indeed the value of DevNonce will be the same every time a new join procedure is done by the end-device. In this case the network server will discard the join request message because it has stored the previous values of DevNonce used by the end device and, in order to prevent replay attacks, it discards the request from that end device with the same DevNonce and the end device is not able to join the network.

5.4.1 Case a) Saturation of the receiver

We first analyze the case a). Through the Friis transmission equation the power received is proportional to the power transmitted as

$$P_t[dBm] = P_r[dBm] - G_r[dB] - G_t[dB] - 20\log_{10}\left(\frac{c}{4\pi f d}\right) \tag{5.3}$$

where G_r and G_t are the receive antenna and transmit antenna gains respectively, $c = 3 \cdot 10^8$ m/s is the speed of light in free space, d is the distance between the two antennas in meters and f is the frequency of radio transmission in Hz. In the LoraWAN protocol the devices use the ISM band (**I**ndustrial, **S**cientific and **M**edical radio band). In Europe this band has frequency between 863 MHz and 870 MHz. So keeping the central frequency $f = 868$ MHz the last term of (5.3) at a distance $d = 1$ m is

$$20\log_{10}\left(\frac{c}{4\pi f d}\right) \simeq -31.2\ dB. \tag{5.4}$$

For example, if the end device is a Waspmote of Libelium Comunicaciones Distribuidas S.L [8][9], the receive antenna gain G_r can be of 0 dB or 4.5 dB. Considering that the dynamic range of the RSSI value is 127 dBm (as reported in [10]), the maximum value of RSSI that can be written in the register is

$$RSSI_{max}[dBm] = S + 127\ [dBm] \tag{5.5}$$

where S is the sensitivity value, that is the minimum value of RSSI that can be measured by the device. In the case of RegRssiValue for example, the sensitivity value S is set to -139 dBm. However in the datasheet the value of S for the RegRssiWideband is not reported, but we can imagine that its value is similar to that of RegRssiValue. In every case, considering the gain of antennas $G_r = G_t = 0$ dB, with an end-device transmitting at a distance of 1 meter with power $P_t = 14$ dBm, the received is

$$P_r \simeq 14 - 31.2 \simeq -17.2\ dBm \tag{5.6}$$

and it is possible to saturate the receiver if

$$S = -17.2 - 127 = -144.2\ dBm. \tag{5.7}$$

This value is probable too small with respect of that set in the device. However if the transmitting device is a distance $d = \lambda = \frac{c}{f} \simeq 34,5$ cm the attenuation of the channel is

$$20\log_{10}\left(\frac{1}{4\pi}\right) \simeq -22\ dB \tag{5.8}$$

and the received power is $P_r \simeq -8\ dBm$. So in this case

$$S = -8 - 127 = -135\ dBm \tag{5.9}$$

that is a possible value of S. Even if the value of S is higher, due to random phenomena such as thermal noise, fading, etc. the receiver power sometimes is higher than that evaluated through the Friis equation and the probability to saturate cannot be negligible.

Unfortunately, in this analysis we don't have consider that the SX1272 transceiver has a LNA with an AGC (Automatic Gain Control), that attenuates the signal with high power (see Tab. 4.1). In particular the higher attenuation introduced is -48 dB. So considering also the contribution of LNA the $RSSI_{max}$ value is difficult to achieve. However we have also to consider the response time of the LNA: if a signal alternatively transmit with high and low power, the saturation of the RSSI value is possible for few instants. Furthermore it is also possible to obtain 0 values for few instant if, after an high power signal transmission, we transmit a low power signal. We examine in the next chapter a situation where the maximum and minimum values of RSSI are written.

5.4.2 Case b) Constant value of RSSI

Finally we have to examine the case b), that is the receiver doesn't saturate but the value of RSSI doesn't depend on noise. Since the RegRssiWideband register writes value of RSSI is quantized with a step of 0.5 dBm, i.e. a non-linear quantization is performed, the more is the received power the more power is needed to go from one power level to an adjacent one. So, theoretically, if an high power is received, the value written in the RSSI may not depend on noise. However, through the equation of RSSI variance in (4.16) , we can observe that the variance is directly proportional to the power voltage, as it is for the size of levels using a logarithmic scale. Indeed supposing to have the quantized value R (in dBm) of RSSI, the size of the R^{th} level, in linear scale, is

$$\begin{aligned} \text{size}[R^{th}\ \text{level}] &= 10^{(R-30+0.25)/10} - 10^{(R-30-0.25)/10} = \\ &= 10^{R-30}(10^{0.025} - 10^{-0.025}) = R_W \cdot L\ [W] \end{aligned} \tag{5.10}$$

where the factor -30 has been inserted in order to have the value of size in watt, R_W is the value written in the register in watt and $L = 10^{0.025} - 10^{-0.025} \simeq 0.115$ is the factor of proportionality between the value of RSSI and the size of that level. So the choice of a logarithmic scale for the quantization of RSSI seems to be coherent with the equation of variance of RSSI in (4.16),

that is since variance is directly proportional to the voltage power, it is reasonable to choose a non-uniform size for the quantization levels, but a logarithmic scale that is also proportional to the voltage power.

However, observing Figg. 4.9, 4.10 and 4.11, we can notice that sometimes there are few values of RSSI that are obtained with a non-negligible probability and, especially, one value is obtained with an high probability ($> \frac{1}{2}$). Considering the step of 0.5 dB we have

$$\begin{aligned} P[0] &= Pr[mod(RSSI, 1) = 0] \\ P[1] &= Pr[mod(RSSI, 1) = 0.5] \end{aligned} \tag{5.11}$$

where $mod(RSSI, 1)$ is the remainder after division (in this case is the fractional part). This quantization of the distribution of RSSI can bring sometimes to a non-uniform probability to generate a 0 or 1 bit. Taking, for example, the plot of pmf in Fig. 4.10 with $\frac{v}{\sigma_w} = 10$ and $K = 64$ we have

$$\begin{aligned} P[0] &\simeq 0.9 \\ P[1] &\simeq 0.1. \end{aligned} \tag{5.12}$$

This is the worst case scenario that we have analyzed. Analyzing, instead, the best case scenario, that is $K = 1$ and only nois voltage we obtained

$$\begin{aligned} P[0] &\simeq 0.5 \\ P[1] &\simeq 0.5. \end{aligned} \tag{5.13}$$

However, not knowing some parameters of the receiver (such as the number of samples used for the evaluation of RSSI), we don't know which are the theoretical pmf's that is possible to achieve. Moreover in this analysis we haven't consider the presence of noises or signals that can't be model as a Gaussian random variable. In the next chapter we will analyze the experimental RSSI pmf's in several situations, in order to evaluate if the theoretical model is correct and also if the probability to generate a bit is uniform. Furthermore we will also analyze the experimental pmf of DevNonce, in order to evaluate if the distribution is quite uniform.

Chapter 6

Experimental results

In this chapter we analyze experimentally the values of RSSI written in the register RegRssiWideband and DevNonce generated by an end-device, equipped with SX1272 transceiver, in a domestic environment. The device used for the experiments is a WiMOD SK-iM880A [18]. In particular we observe if the recommended procedure for the generation of random numbers [6] works well in these situations:

- without any device transmitting in the proximity of the WiMOD;
- putting the receiver inside a metal box;
- with another WiMOD transmitting in LoRa mode at a distance of 1 m;
- with another WiMOD transmitting in LoRa mode or OOK mode at a distance of $\lambda \simeq 35$ cm.

Let's see in the next sections, case by case, what we have obtained.

6.1 WiMOD without jammer

In this situation the RSSI value obtained should concern thermal noise or interfering signal of the environment. We collected 891840 values. In Fig. 6.1 we reported the pmf of the values.

First of all, the values in x-axis is the integer number written in the register, that doesn't correspond with the value of RSSI but it is linked through the equation

$$RSSI = S + \frac{R}{2} + A_{AGC} \; [dBm] \tag{6.1}$$

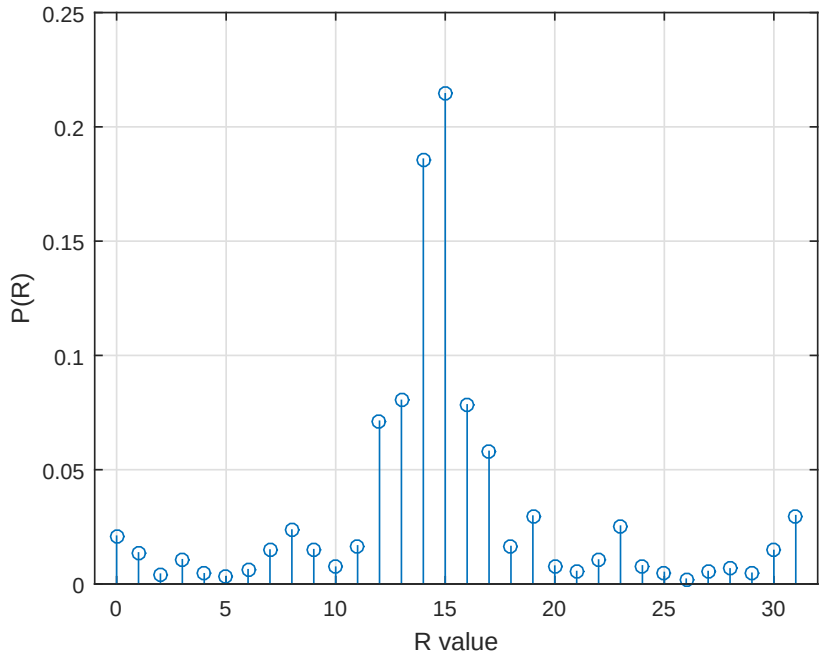

Figure 6.1: Pmf of the R values obtained without any device transmitting in the proximity of the WiMOD

where $RSSI$ is the value of RSSI that arrived at the receiver and attenuated by the AGC, S is the sensitivity level, A_{AGC} is the attenuation introduced by the AGC and R is the value written in the register.

From Fig. 6.1 we can observe a trend that may be compatible with quantized noncentral chi-squared random variable, even if there are secondary lobes that may be due to interfering signal. In particular, comparing the experimental values with theoretical pmf's in Fig. 4.9, observing the probability of the most probable value (about 0.2), we have a similar value of probability with $K = 32$ or $K = 64$.

For the generation of random numbers, we are interested in the probability of generating a 0 bit and a 1 bit, and also in the probability to generate the sequence '00','01','10','11' in order to evaluate also the correlation on the bit generation. We reported the obtained probability in Tab. 6.1.

We can observe that the probability to generate a 1 bit and a 0 bit is slightly unbalanced and it is more probable to generate a 1 bit instead of a 0 one. This result is in contrast to that obtained theoretically: indeed in case of only noise voltage, for value of K from 1 to 64 we obtain a uniform probability on generation of single bit. So in this case the displacement can

Parameter	Value
# of collected values	891840
$P[0]$	0.468
$P[1]$	0.532
$P[00]$	0.219
$P[01]$	0.249
$P[10]$	0.249
$P[11]$	0.283
Most probable value	15
Maximum probability	0.21

Table 6.1: Probability of generating the specified sequences of bits and other parameters without jammer.

due to secondary lobes and saturation (both under the sensitivity level and over the maximum level). Indeed a characteristic aspect that we observed is that, when there aren't devices transmitting, the maximum written value is 31, as if the designers decide to use only 5 bits to register the value, when no signals arrive. So, in this situation, the value 31 should be considered as the saturated value.

Observing instead the other probabilities two distinct generation of a bit seem to be independent. Indeed, if two consecutive generations of a bit are independent

$$\begin{aligned} P[00] &= P[0]^2 = 0.468^2 = 0.219 \\ P[01] &= P[0] \cdot P[1] = 0.468 \cdot 0.532 = 0.249 \\ P[10] &= P[1] \cdot P[0] = 0.532 \cdot 0.468 = 0.249 \\ P[11] &= P[1]^2 = 0.532^2 = 0.283 \end{aligned} \tag{6.2}$$

and these values are equal (with approximations) to that obtained experimentally.

We have also observed the trend of pmf with time, in order to evaluate if the behavior is time-varying. The procedure is to consider one thousand consecutive samples of R and plot the pmf of the values. We have observed

that the pmf doesn't change behavior with time. In particular the most probable value remains 15 (rarely is 14) and secondary lobes are always present (see Fig. 6.2).

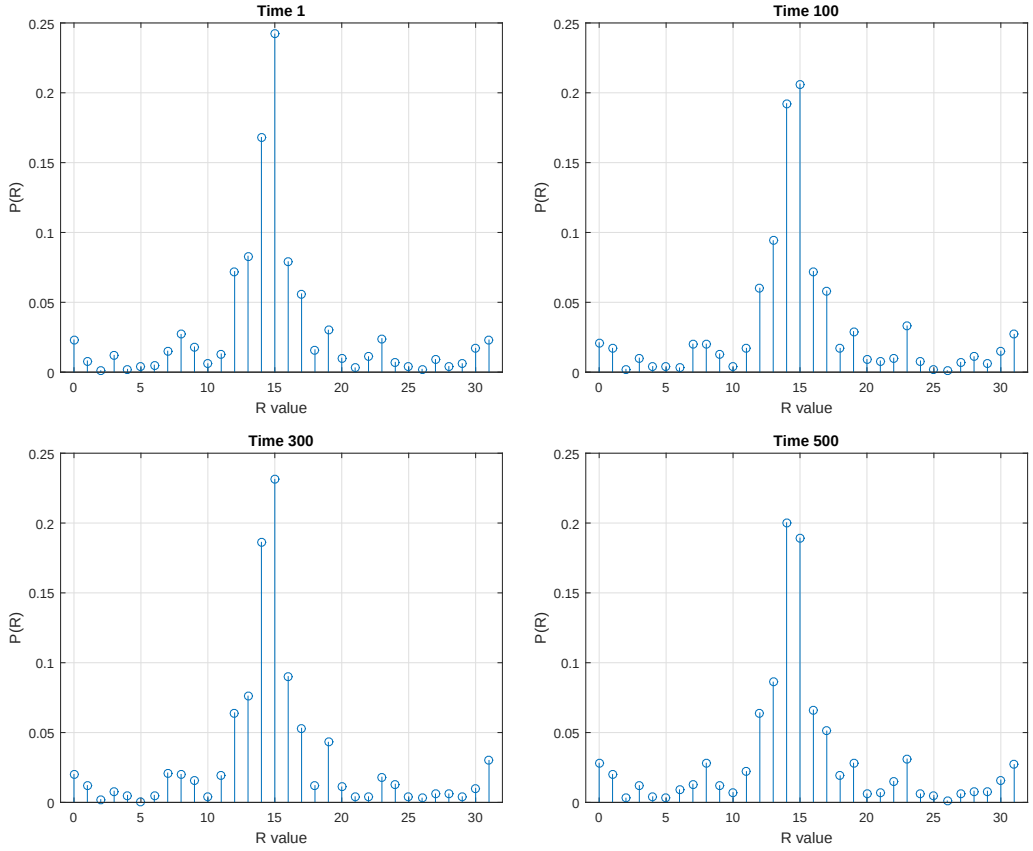

Figure 6.2: Dependence of R pmf with time.

Let's now examine how the unbalanced probabilities influence the generation of DevNonce. We collected one milion sample of DevNonce in order to have on average 15 generated DevNonce for each value. Two important parameters for a random number generator are *entropy* and *min-entropy*, defined respectively in (5.1) and in (5.2). In Tab. 6.2 we sum up the most important features. Even if the probability to generate a bit is slightly unbalanced the procedure reaches an entropy that is close to the theoretical bound, 15.9 bits instead of 16; so, on average, every number is repeated every $2^{15.9} \simeq 61147$ times. However, for the generation of random numbers, as seen in the previous chapter, it is more important the parameter called min-entropy. In this realization we obtained the value 14.2 bits. Having a min-entropy of 14.2 bits means that the most probable value appears on aver-

age every $2^{14.2} \simeq 18820$ generations instead of $2^{16} = 65536$. Since the number of generated DevNonce in the life of a LoRa end-device is similar to that estimated in Chapter 3, on average the most probable value is regenerated after 2.6 lives ($\frac{18820}{7300}$).

Finally we have also evaluated the averaged 1^{st} regeneration of a DevNonce, defined as the first value that has been already generated. In order to have an averaged value with one realization, we divide the collection in subsets of size $S = 7300$. The size has been chosen considering the analysis performed in section 3.2. In this manner the value is evaluated averaging N_{subset} regeneration values, with

$$N_{subset} = \lfloor \frac{\#\text{ of collected values}}{S} \rfloor. \tag{6.3}$$

For example, with 10^6 collected values, we have $N_{subset} = 136$. We can observe that the averaged value in Tab. 6.2 is almost equal to that evaluated theoretically in (3.8) with a uniform probability of bit generation.

Parameter	Value
# of collected values	1000000
Entropy	15.90
Min-entropy	14.20
Most probable value	49151
Least probable value	4357
1^{st} regeneration	320.6

Table 6.2: Characteristics of collections of DevNonce without a jammer.

The presence in the R distribution of secondary lobes that are not present in the theoretical distributions requires to verify if there are interference in the environment, that brings these results. For this reason, in order to isolate the system from the environment as much as we can, in the second case we put the device in a metal box (that is a Faraday cage).

6.2 WiMOD in a metal box

In order to evaluate if the secondary lobes are due to interfering signal in the environment, we put the device inside a metal box, in order to isolate the system from electromagnetic waves as we can. Unfortunately we don't have possibility to put the WiMOD inside an anechoic chamber. As in the previous case we evaluate the probability of generating $0, 1, 00, 10, 01, 11$ (reported in Tab. 6.3). Fig. 6.3 reports the pmf of R values.

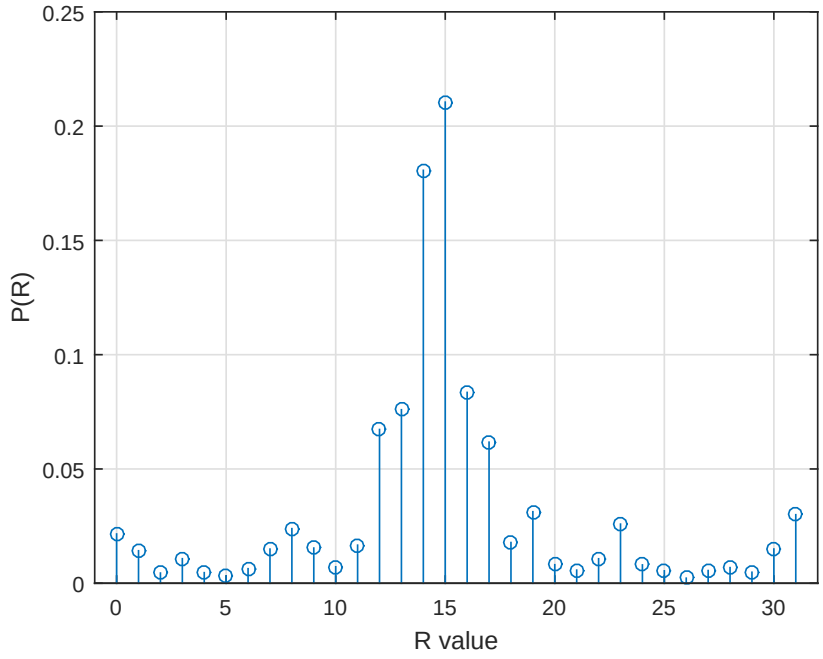

Figure 6.3: Pmf of the R values obtained putting the WiMOD inside a metal box.

Also in this case the most probable value is 15 and secondary lobes are present. Then we conclude that, probably, secondary lobes are not caused by external signals. Then we suppose that secondary lobes are caused by electromagnetic waves or other non-thermal (and non-Gaussian) noise generated by the end-device. However, another possible hypothesis is that the pmf is influenced by AGC and its ability to attenuate the power. Indeed in (6.1) the same value of R can be achieved with two or more different values of RSSI if the level of attenuation A_{AGC} is different. So, since in the theoretical analysis we have considered the distribution of RSSI, while in the experimental results we plot the pmf's of R, it is evident that the distributions are

different.

Moreover, in the metal box, the probabilities in Tab. 6.3 are equal to that reported in Tab. 6.1, with unbalanced probability to generate a 1 bit.

Parameter	Value
# of collected values	998453
$P[0]$	0.468
$P[1]$	0.532
$P[00]$	0.219
$P[01]$	0.249
$P[10]$	0.249
$P[11]$	0.283
Most probable value	15
Maximum probability	0.21

Table 6.3: Probability of generating the specified sequences of bits and other parameters with WiMOD in a metal box.

Let's now examine the generation of DevNonce through Tab. 6.4.

Parameter	Value
# of collected values	1481487
Entropy	15.92
Min-entropy	14.50
Most probable value	57327
Least probable value	2325
1^{st} regeneration	312.7

Table 6.4: Features of DevNonce's collection with WiMOD in a metal box.

Also for DevNonce, the results are similar to those obtained in the previous section, in particular the values of entropy and min-entropy. We can also observe that in both situations the least probable values are values containing more 0 bits than 1 bits (2325 has 5 ones and 11 zeros, 4357 has 4 ones and 12 zeros) and conversely the most probable values contained more 1 bits (49151 has 15 ones and 1 zero, 57327 has 14 ones and 2 zeros), reflecting the probabilities obtained for the generation of zeros and ones with R.

Finally, also with the WiMOD in the metal box, we obtain a value of 1^{st} regeneration congruent with that evaluated in (3.8).

6.3 WiMOD with jammer at a distance of 1 m

Even if also without a jammer we have seen that the probability to generate a 1 bit is slightly higher than that of generating a 0 bit, we want to analyze if the value of R can be made more biased, using a jammer. In this case, if the transmitted signal is constant, we can increment the ratio $\frac{v}{\sigma_w}$ obtaining a larger displacement on the probability to generate a bit (as seen in the previous chapter). In our case the jammer is another WiMOD SK-iM880A, that is transmitting in LoRa mode with power of 14 dBm. We collected the values of R in three different time instants and the values of DevNonce in other two instants. Indeed in a domestic environment the channel is time varying. Moreover we collected the values in different rooms and in some cases we transmit a random message, while in others we transmit a string of ones. Then the collected data present some differences, that sometimes are difficult to explain theoretically. Furthermore, due to the particular implementation, we transmit finite messages and from one message to the following we have a break of the transmission, that causes anomalies on the collected values.

Let's firstly examine the values of R. In Tab. 6.5 we reported the probabilities explained beforehand. The results can be also understood observing Fig. 6.4. Also in this case, two consecutive generations of a bit seem to be independent. However, with a jammer, the displacement between $P[0]$ and $P[1]$ is incremented, in particular in the second experiment. The fact that in the first and in the second experiment $P[0] > P[1]$, while in the third one is the opposite, is probably due both to distance imprecision and differences on the environment, that change the most probable value of R.

Furthermore from Fig. 6.4 we can observe that in some circumstances, more than one peak are present. The first peak around the value 15 is due to the fact that the jammer periodically stops the transmission, so at the

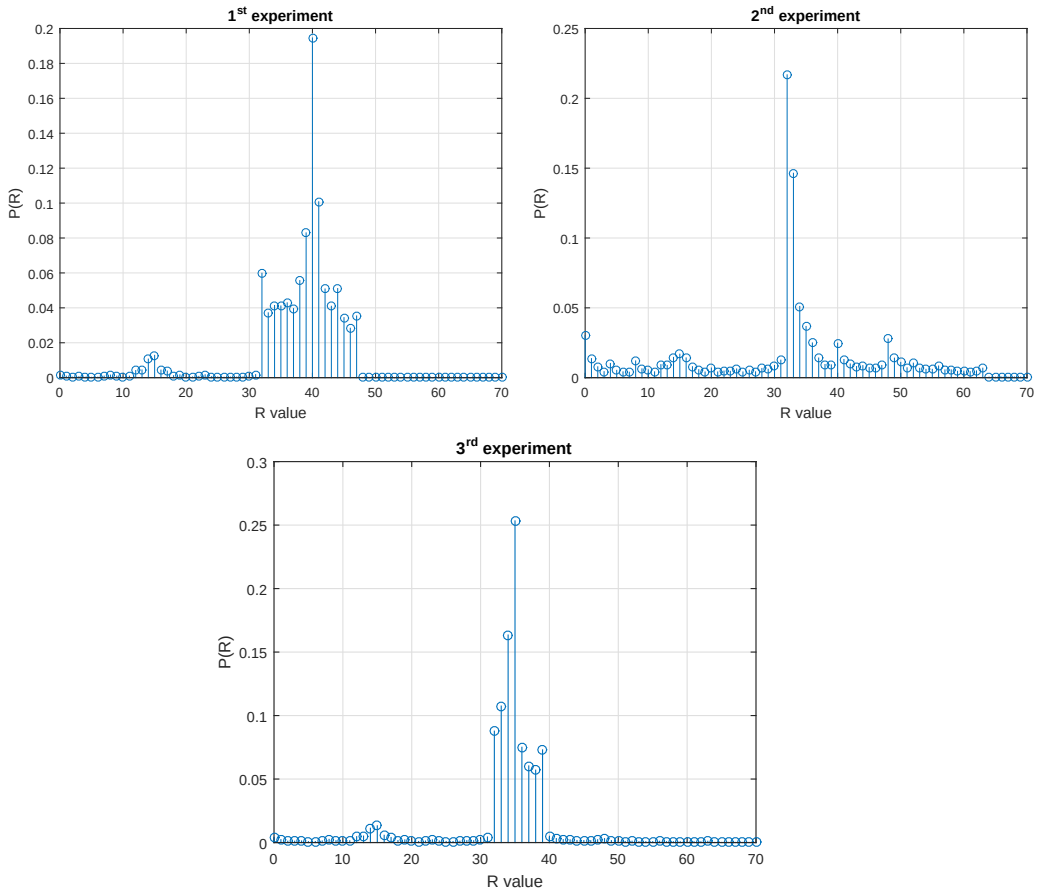

Figure 6.4: R values with a jammer at a distance of 1 meter.

WiMOD, that is measuring R, the jamming signal doesn't arrive and the value of R is due only to noise. An interesting aspect that we have observed is that, after the stop of the transmission, the following value measured of R, sometimes, is very high (see Fig. 6.5). As we will demonstrate in the next chapter, this behavior is motivated by the presence of LNA and automatic gain control (AGC): when there aren't electromagnetic waves arriving at the device, the LNA is set with maximum gain; at the beginning of transmission the gain of LNA is still set with the maximum gain and it is not attenuating the strong signal, since the response time is not instantaneous and the value written in the register is near the saturation value. A first evidence of this hypothesis is that we never measured two consecutive high values: indeed since the saturation is due to a delayed response in the AGC, after a while the measured values of R will be far from the saturation level because, as we have seen theoretically in subsection 5.4.1, with another LoRa device

Parameter	1^{st} experiment	2^{nd} experiment	3^{rd}experiment
Message of jammer	not known	string of 1's	string of 1's
# of collected values	102720	999998	2000000
$P[0]$	0.554	0.587	0.446
$P[1]$	0.446	0.413	0.554
$P[00]$	0.309	0.346	0.203
$P[01]$	0.245	0.241	0.243
$P[10]$	0.245	0.241	0.243
$P[11]$	0.201	0.172	0.311
Most probable value	40	32	35
Maximum probability	0.20	0.22	0.25

Table 6.5: Probability of generating the specified sequences of bits with a jammer at a distance of 1 meter.

(that is with transmission power of 14 dBm) is almost impossible to saturate the receiver when the maximum level of attenuation is set. However, since this phenomenon is due to received power difference between the periods of transmission and the periods of silence, we will examine in depth this behavior in the next section, where, putting the jammer closer to the receiver, the phenomenon will be enhanced.

Regarding the probability of the most probable value, only in the third case we have obtained a value slightly higher than that with only noise, while in the other cases the results are similar to that of previous scenario. These results seem to be compatible with the quantized pmf of noncentral chi-squared distribution with a low ratio $\frac{v}{\sigma_w}$, as if the task of the AGC is that of maintaining small this ratio.

Another important aspect is that with a not known message we have obtained a broader peak than those obtained with a string of ones: this is probably due to the higher entropy of the message, that is translated with an higher variance of the amplitude of the transmitted electromagnetic wave, and so of the received power. Furthermore when the message has low entropy, the AGC works better and, on average, the measured values of R is lower

221115	34
221116	27
221117	40
221118	14
221119	251
221120	25
221121	20
221122	32
221123	33
221124	35
221125	32
221126	41
221127	32
221128	49
221129	32
221130	32
221131	4
221132	50
221133	34
221134	33

Figure 6.5: Consecutive measurements of R value. We can observe that after a low value (14) we obtain a value closer to the saturation value (251).

($32 \div 35$ against 40). Finally we have also observed that the initial measured values of R is slightly higher than the average value, as if initially the AGC less efficiently attenuates the high power signals.

Let's now examine the generated values of DevNonce. As said before we collected the values in two different moments. In the first case the jammer has transmitted an unknown message, while in the second the message is a string of ones. In Tab. 6.6 we sum up the most important aspects of the collections.

Also with a jammer at a distance of 1 m the entropy and min-entropy values are similar to that without a jammer, sometimes even better. In particular the best values are achieved when a string of ones is transmitted, that also corresponds with a longer observation time. So considering the time varying channel and other random phenomena, it is probable that the longer is the observation time, the higher is the entropy.

Concerning the last parameter in Tab. 6.6, instead, we can observe that the slightly increment of the non-uniformity on the bit generation probability brings a slightly decrease on the value of 1^{st} regeneration, with respect of that evaluated theoretically, even if the reduction is not so relevant (around 5%).

However, at a distance of 1 meter, other phenomena that are different from thermal noise seem to be still relevant. For this reason, in the next section, we set the jammer closer to the receiver, in order to decrease the contribution of this phenomena.

Parameter	1^{st} experiment	2^{nd} experiment
Message of jammer	not known	string of 1's
# of collected values	991849	1685034
Entropy	15.87	15.95
Min-entropy	13.87	14.64
Most probable value	65535	81
Least probable value	149*	64828
1^{st} regeneration	304.8	304.3

Table 6.6: Characteristics of collections of DevNonce with jammer at a distance of 1 meter.*There are other values with the same probability.

6.4 WiMOD with jammer at a distance of λ

In order to better verify if the theoretical analysis is coherent with experimental results we set the jammer at a distance of $\lambda \simeq 35$ cm in order to reduce phenomena as fading, reflections, etc. In this case we collected the values in three different moments with different settings. In the first case we have transmitted in LoRa mode a not known message; in the second we have transmitted in LoRa mode a string of ones; finally in the third experiment we have transmitted in OOK mode a string of ones.

The results obtained in this case are more different from each other. For example in the first case we obtain a quite uniform probability to generate a 0 or a 1 bit, due probably to the variance of the message.

Furthermore, as seen in previous sections, since the variance of the message is higher, the AGC works with low efficiency and we obtain a relatively high value of R with respect of the other two experiments (75 against $36 \div 40$). Moreover if the variance of the message is small the most probable value of R is quite similar to that obtained at a distance of 1 m in the same setting. This confirms the hypothesis that AGC try to set the receiver in the same condition (of variance and mean), independently on the received power; but if the variance of message is relevant the AGC has more difficult to set the receiver in the 'standard condition'.

In the second experiment, instead, we obtained an unbalanced probability of generation of bits, with values similar to that obtained with the jammer

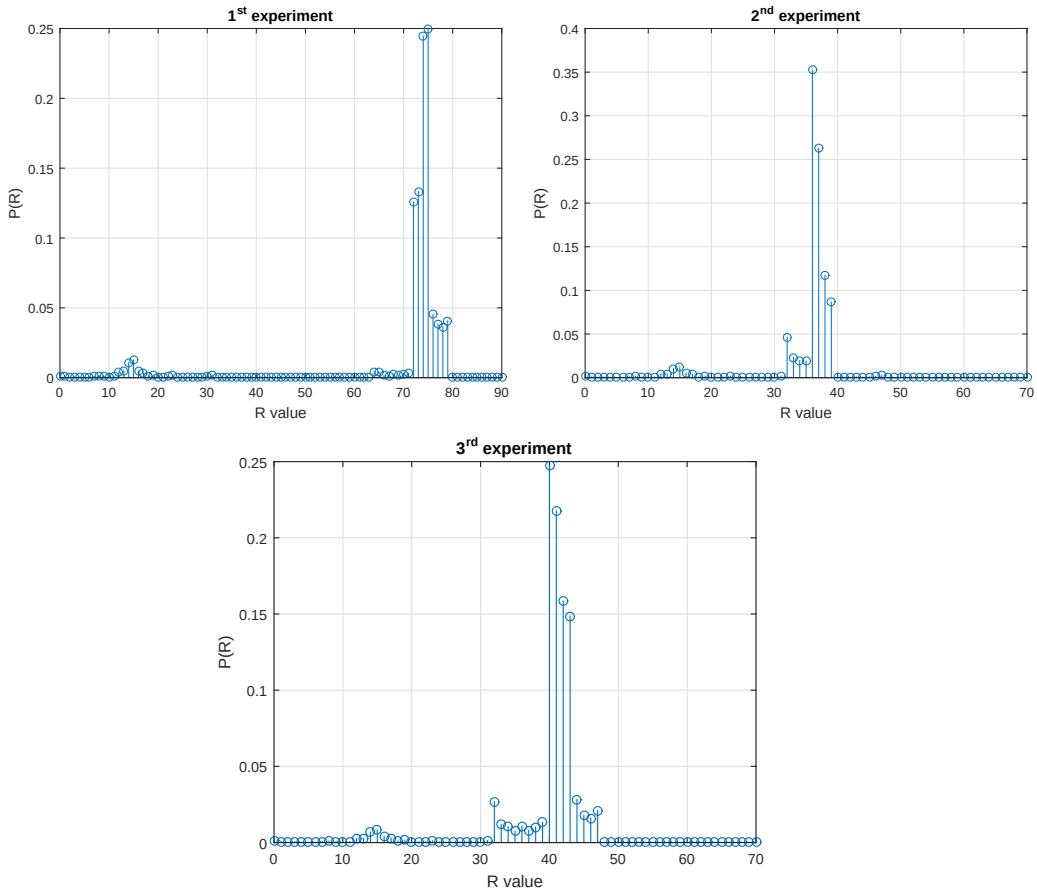

Figure 6.6: R values with a jammer at a distance of λ meter.

at a distance of 1 meter. However the most relevant result that we have achieved is that the most probable value has a considerable increment of the probability (0.35 against 0.25), as if in this case the ratio $\frac{v}{\sigma_w}$ is higher than previous settings and conditions.

In order to still increment the ratio $\frac{v}{\sigma_w}$, in the third experiment we set the OOK modulation, transmitting a string of ones, in order to have a constant received power as possible as we can. Unfortunately in this case we don't have obtained the expected results. First of all the probability of generating a 0 bit and the probability of generating a 1 bit are consistent with the results obtained without a jammer, so we don't increment the probability of the most probable value (as seen theoretically incrementing the ratio $\frac{v}{\sigma_w}$). An hypothesis is that with a message without entropy, the AGC is able to reduce adequately the power of the received signal, in order to have the preferable ratio $\frac{v}{\sigma_w}$. Moreover, since the received power is constant, the AGC doesn't

Parameter	1^{st} experiment	2^{nd} experiment	3^{rd} experiment
# of collected values	992525	2000000	2000000
Type of modulation	LoRa	LoRa	OOK
Type of message	not known	string of ones	string of ones
$P[0]$	0.494	0.569	0.530
$P[1]$	0.506	0.431	0.470
$P[00]$	0.244	0.325	0.283
$P[01]$	0.25	0.245	0.248
$P[10]$	0.25	0.245	0.248
$P[11]$	0.256	0.186	0.222
Most probable value	75	36	40
Maximum probability	0.25	0.35	0.25

Table 6.7: Probability of generating the specified sequences of bits and other parameters with a jammer at a distance of λ.

have to change level of attenuation and we can obtained conditions that are more stable even of that without a jammer.

Another important aspect that we have already observed with the jammer at distance of 1 m and that now is more evident, is the saturation of the R value. As we can see from Fig. 6.8, after a low value of R (12) in the register is written the maximum value. As outlined in the previous section, the jammer periodically stops the transmission for a short period of time. We have hypothesized that the stop in the transmission, that we will call *silence period*, is due to the implementation of the jammer software. Indeed the jammer periodically transmit a 10 byte message, through a while cycle and between the n^{th} and $n^{th} + 1$ cycle there is a transmission break. In order to verify if the hypothesis is correct we transmit 1000 messages and we evaluate if we have 1000 silence periods. We mark as a silence period a value of $R < 32$, since in section 6.1 the value of R never exceeds this bound. Moreover, observing all the distributions of R with the jammer at distance of 1 m and at distance of λ, there is always a notable difference between $P(31)$ and $P(32)$, as if there is a threshold between these values that marks the

values due to noise from that due to received signal. However sometimes we have observed two consecutive values under 32 (but never more than two): in this case we evaluate these values as a single silence period. With this procedure we have obtained exactly 1000 silence periods and our hypothesis is verified. Moreover we always have measured a low value of R every 26 or 27 values of R, excluding, obviously, the consecutive values (see Fig. 6.7).

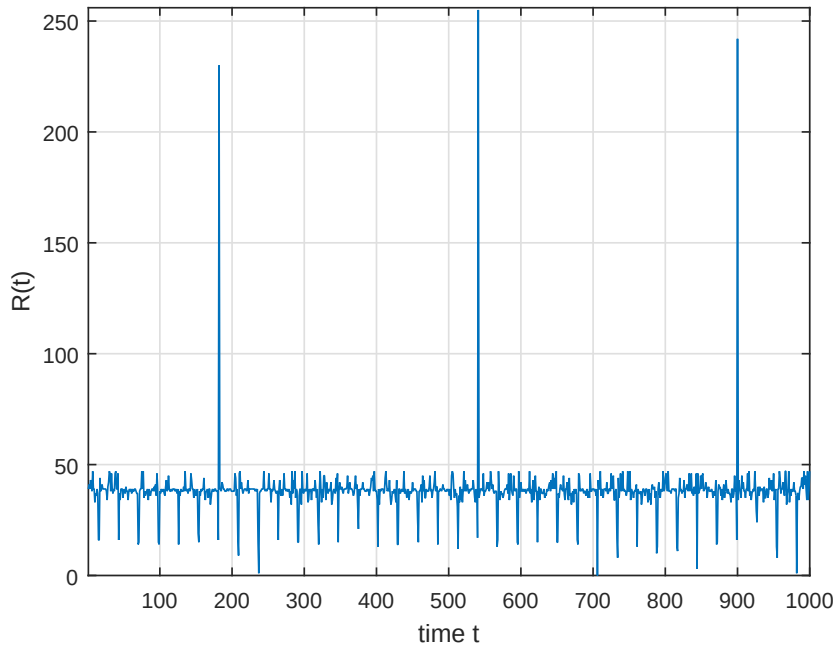

Figure 6.7: Trend of R with time. We can observe the periodicity of low values.

Then, in order to proof that the high values of R is due to a delayed response time of the AGC we have counted how many values greater or equal to 200 are registered after a silence period, that is after a value smaller than 32. However we have only consider the collection where we have transmitted a string of ones in LoRa mode. Indeed, in the simulation with unknown message, we have observed more different behaviors, due probably to the variance of the message, while, in OOK mode, we have measured few high values. So at distance of 1 m, summing the results of second and third experiment, we have founded 212 values greater or equal to 200 and all these values has been registered after a silence period. At a distance of λ we have observed the same behavior, that is we have counted 5683 values greater or equal to 200 and all of these values arrived after a silence period. Moreover

33 of these values correspond with the saturation value (255).

However, even if high values of R is due to a delayed response time of the AGC, from Fig. 6.7 we can observe that we don't always have an high value of R after a silence period. We have motivated this feature with the fact that R is an averaged value. Indeed, in order to obtain an high value of R, the majority of the averaged samples of a single measurement of R must be high: this means that the samples of a single value of R must be aligned with the beginning of the transmission (and simultaneously the AGC must not have update its attenuation level). If, for example, the value of R is evaluated averaging 32 samples and the first 20 samples are measured during the silence period, while the remaining ones are measured when the jammer is working, we don't obtain an high value of R. This hypothesis is confirmed through the trend of R with time in Fig. 6.9: we can observe that the trend presents a periodicity in time, and also the high values are obtained periodically. This means that, periodically, the samples of a single measurement of R are aligned with the beginning of transmission and an high value of R is obtained. In Fig. 6.10 we can see better the values over 250 and under 32.

Furthermore, during a silence period, we can also achieve a value of R that is under the sensitivity level and a 0 value can be written. Indeed, a delayed response time of the AGC can also permit to write a value under the sensitivity level: if the AGC is strongly attenuating an high power received signal, that later disappears, in the register will be write a 0 value with a consistent probability, because the AGC is strongly attenuating a signal due to noise (that is very weak). However also in this case there must be the alignment of silence period with the samples of a single measurement of R. Considering the experiments where we have transmitted a string of ones in LoRa mode, we have founded 28418 zero values after a value larger than 31 out of 38859 zero values (73%) in the experiments with jammer at a distance of 1 m, while 1475 zero values after a value > 31 out of 2482 zero values (60%) at a distance of λ.

This behavior suggests us a possible attack that writes alternatively a value under the sensitivity level and a saturated value. Indeed if we are able to transmit pulses with appropriate power, duration and gap it may be possible to periodically saturate the receiver when the pulse is transmitted, and then to record a 0 value during the silence period. In this manner the pulses causes a generation of a 0 bit (given by the value $R = 0$) followed by a 1 bit (given by $R = 255$), and vice versa. In our experiments we have counted a low number of saturated values and a low number of 0 values, but considering that these results were unexpected, probably through a specific procedure, may be possible to obtain more saturated and 0 values.

988476	72
988477	76
988478	72
988479	12
988480	255
988481	74
988482	72
988483	65
988484	75
988485	73
988486	72
988487	72
988488	72

Figure 6.8: Consecutive measurements of R value. We can observe that after a low value (12) we have the saturation of the R value (255).

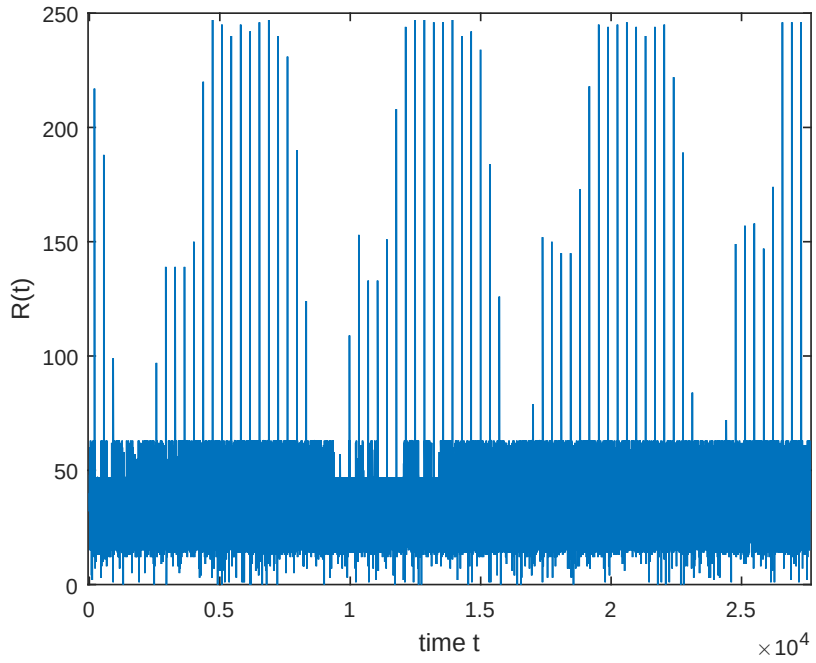

Figure 6.9: Trend of R with time. We can observe the periodicity of high values.

Let's now analyze the DevNonce generated in the same three settings used for the R.

In the first case, where the message sent is not known, the results are similar to that with a jammer at a distance 1 m and that without a jammer. Indeed we reach an entropy closer to the theoretical bound and a value of

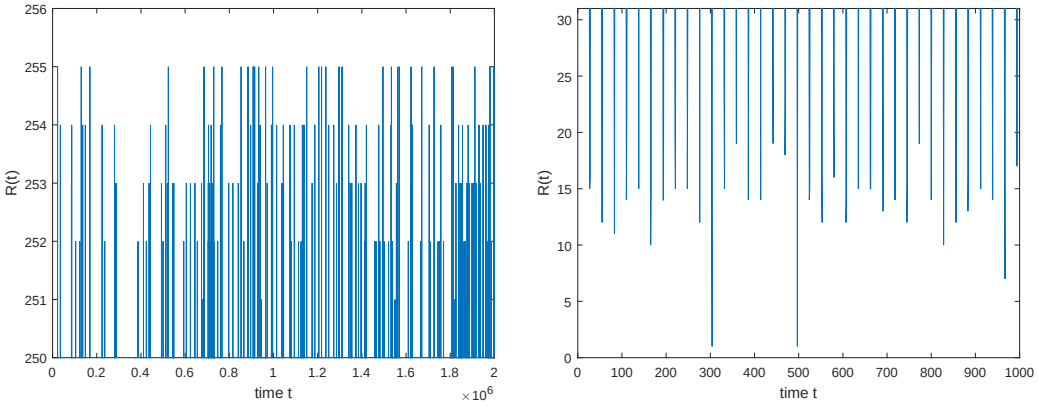

Figure 6.10: R values over 250 and under 32.

Parameter	1^{st} experiment	2^{nd} experiment	3^{rd} experiment
# of collected values	1582641	2000000	2000000
Type of modulation	LoRa	LoRa	OOK
Message of jammer	not known	string of ones	string of ones
Entropy	15.94	15.84	15.97
Min-entropy	14.10	12.66	15.07
Most probable value	0	65535	352
Least probable value	44156	14789	13184
1^{st} regeneration	293.8	271.1	321.3

Table 6.8: Characteristics of collections of DevNonce with jammer at a distance of λ meters.

min-entropy consistent with the other values of min-entropy obtained in the previous situations. However the fact that the most probable value is 0 seems to indicate that the probability of generation of a bit is biased with $P[0] > P[1]$, as the least probable value contains more 1 bits than 0 bits, while, in the same situation, in the collection of R we measured $P[0] \simeq P[1]$.

In the second case, instead, the most relevant value is the min-entropy, that is lower than in the other cases. A min-entropy of 12.66 means that the most probable value is generated, on average, every 6451.6 procedures.

In the third case we achieve the best conditions for the generation of DevNonce. Indeed we obtained the highest measured values of entropy and min-entropy, respectively 15.97 and 15.07 bits. These results are consistent with the hypothesis that the AGC is able to set the best conditions for the generation of random numbers (i.e. low ratio $\frac{v}{\sigma_w}$) with the level of attenuation that doesn't change since the received power is constant.

However, comparing the parameters obtained in the second experiment, we have to discard the hypothesis made in section 6.3, i.e. it is not true that a longer exposition entails an higher entropy. Indeed the lowest value of min-entropy has been achieved with the same exposition time of the third experiment and second experiment of section 6.3.

Finally, with a jammer at distance λ, the reduction of the 1^{st} regeneration value is more relevant in the first and second experiment. The decrease is even of 18% in the second experiment (that with low value of min-entropy). In the third experiment, instead, the value of the last parameter in Tab. 6.8 confirms the goodness of generation in this situation, since it is almost equal to that evaluated theoretically in (3.8).

Chapter 7

Conclusion

We can divide the work of this thesis in two parts: the first concerns the study of LoRaWAN™ protocol and the analysis of join procedure; the second is about the recommended method for the generation of random numbers for SX1272, that, at the state of the art, is the only transceiver used in LoRaWAN™ protocol. The two topics are linked by the DevNonce, that is a 16-bit random numbers used in the join request message.

In the first part, after an introduction on LoRaWAN protocol and networks, with particular attention on security mechanisms, we highlighted the problems related with the join procedure, due both to non-deepened aspects and to weaknesses on the protocol. In particular it has been analyzed what is the best number of DevNonce's that network server must record per each end-device; then it has been discussed the convenience for the DevNonce to be random; finally it has been shown a security breach in the protocol related with the join accept message, that is not immune against replay attacks. Furthermore we introduced possible solutions and/or alternatives to the problems.

In the second part, instead, after an introduction on the random number generation and on the architecture of the receiver, we studied theoretically and experimentally the procedure for the generation of random numbers, analyzing its efficiency. If, theoretically, we observed that, in particular situations, the probability of generating a 0 bit can be much different to that of generating a 1 bit, experimentally we obtained partial results, i.e., most of the time, the generation of bits is not-uniform, but the displacement seems to be not so relevant for the generation of DevNonce's. However it is difficult to reach situations studied theoretically, in which the generation of bits is strongly unbalanced, because the transceiver has control systems (as AGC and LNA) that permits to set the receiver in good circumstances for the generation of bits.

However both parts of this work can be further extend. In particular in the first part we focused our attention only on the mechanism of join procedure, but additional weaknesses may be found analyzing the other sections of the protocol. Moreover it is possible to examine in depth the alternatives to DevNonce, such as the employment of a sequential number instead of a random number.

In the second part, instead, the experimental analysis should be extended. Indeed in our analysis we evaluated only some essential parameters in the generation of RSSI and DevNonce. However more specific tests are needed to verify the correctness of the procedures, such described in [19]. Moreover the analysis should be extended to other environments, such as urban environment. Furthermore the observation that after a low value of RSSI, due to a break transmission of jammer, an high value of RSSI is written, suggests to saturate the receiver bypassing the AGC through pulses of established duration, power and gap.

Bibliography

[1] C. Goursaud, J.M. Gorce, *Dedicated networks for IoT : PHY / MAC state of the art and challenges*, EAI endorsed transactions on Internet of Things, 2015.

[2] http://www.semtech.com/wireless-rf/lora/LoRa-FAQs.pdf

[3] N. Sornin, M. Luis, T. Eirich, T. Kramp, O. Hersent, *LoRaWAN™ Specification*, January 2015.[Online]. Available: https://www.lora-alliance.org/portals/0/specs/LoRaWAN%20Specification%201R0.pdf

[4] http://www.radio-electronics.com/info/wireless/lora/lorawan-network-architecture.php

[5] R. Miller, *LoRa Security - Building a Secure LoRa Solution*, MWR Labs. [Online]. Available: https://labs.mwrinfosecurity.com/assets/BlogFiles/mwri-LoRa-security-guide-1.2-2016-03-22.pdf

[6] Semtech, *Recommended SX1272 Settings for EU868 LoRaWAN Network Operation*, January 2015.[Online]. Available: http://www.semtech.com/images/datasheet/an1200.23.pdf

[7] https://github.com/Lora-net/LoRaMac-node/tree/master/src

[8] https://www.libelium.com/contact/#buy

[9] Libelium, *Waspmote Datasheet.*[Online]. Available: http://www.libelium.com/downloads/documentation/waspmote_-datasheet.pdf

[10] Semtech, *SX1272/73 Datasheet.*[Online]. Available: http://www.semtech.com/images/datasheet/sx1272.pdf

[11] S.Antipolis, P. Girard, *Low Power Wide Area Networks security*,December 2015.[Online]. Available: https://docbox.etsi.org/Workshop/2015/201512_M2MWORKSHOP/S04_WirelessTechnoforIoTandSecurityChallenges/GEMALTO_GIRARD.pdf

[12] http://www.link-labs.com/when-should-the-lorawan-specification-be-used/

[13] http://www.link-labs.com/lora-for-control-lighting-locks-and-demand-response/

[14] Joseph J. Carr, *The technician's radio receiver handbook*, Newnes, 2001.

[15] E. Barker, J. Kelsey, *Recommendation for the Entropy Sources Used for Random Bit Generation*, NIST DRAFT Special Publication 800-90B, Second Draft, January 2016. [Online]. Available: http://csrc.nist.gov/publications/drafts/800-90/sp800-90b_second_draft.pdf

[16] D. Eastlake, J. Schiller, S. Crocker, *Randomness Requirements for Security*, June 2005. [Online]. Available: https://tools.ietf.org/html/rfc4086

[17] Wikipedia, *Random Number generation.* [Online]. Available: https://en.wikipedia.org/wiki/Random_number_generation

[18] http://www.wireless-solutions.de/products/starterkits/sk-im880a.html

[19] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, S. Vo, *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*, NIST DRAFT Special Publication 800-22 Revision 1a, 2010. [Online]. Available: http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-22r1a.pdf

Ringraziamenti

Voglio innanzitutto ringraziare i miei genitori per l'affetto dimostrato, per l'educazione e i principi che mi hanno insegnato, per il supporto morale ed economico che mi hanno sempre fornito e per essere sempre stati un punto fisso a cui ancorarmi nei momenti difficili della mia vita. Un ringraziamento inoltre a mia sorella con cui ho avuto un rapporto il più delle volte non idilliaco (come tutti i fratelli) ma a cui comunque voglio un gran bene. Complimenti, inoltre, a te e a Michel per aver dato alla luce quel stupendo e iperattivo ometto di nome Gioele, a cui voglio un mondo di bene. Ringrazio inoltre i miei nonni, i miei zii e miei cugini per l'affetto dimostrato nei miei confronti e per le bellissime giornate vissute insieme.

Ringrazio poi chi mi ha dato la possibilità di svolgere questo lavoro e di portarlo a termine. Ringrazio quindi i professori Stefano Tomasin e Lorenzo Vangelista per l'opportunità che mi hanno dato e per avermi guidato nello sviluppo e nel compimento di questo lavoro, fornendomi numerosi consigli, preziose informazioni e costante supporto. Un ringraziamento, inoltre, a Patavina Tech per avermi fornito i dispositivi su cui lavorare e svolgere la parte sperimentale del lavoro; in particolare voglio ringraziare Ivano che, oltre ad avermi dato i dispositivi, mi ha fornito tutto il materiale che mi era necessario per svolgere gli esperimenti e mi ha aiutato a risolvere i problemi avuti, oltre ad avermi dispensato numerosi consigli e informazioni.

Ringrazio la splendida compagnia di amici con cui sono cresciuto, con cui ho condiviso la maggior parte della mia vita e da cui ho imparato molte cose che sui libri non insegnano. Ringrazio anche le vostre famiglie con cui ho sempre avuto un buonissimo rapporto. Ringrazio i miei amici e compagni delle superiori con cui ho condiviso gioie e dolori e miei compagni dell'università con cui ho condiviso, invece, dolori e gioie. Ringrazio inoltre Casa Grilli, nonché tutti i coinquilini avuti in questi quattro anni vissuti a Padova, con cui ho semplicemente condiviso tutto degli ultimi miei quattro anni di vita e con cui ho passato dei momenti indimenticabili. In generale ringrazio tutte le persone che, anche per poco, hanno condiviso un momento della mia vita e che sicuramente mi hanno insegnato e regalato qualcosa.

Infine voglio ringraziare la persona che negli ultimi mesi mi ha reso felice e ogni giorno contribuisce a rendermi ancora più felice, che mi è sempre a fianco e mi sopporta, che mi regala emozioni uniche e che nei momenti e nelle scelte più difficili riesce sempre a farmi ragionare e a darmi il consiglio giusto.