Security Analysis of LoRaWAN™ Join Procedure for Internet of Things Networks

Stefano Tomasin, Simone Zulian and Lorenzo Vangelista*
Department of Information Engineering, University of Padova, Italy
tomasin@dei.unipd.it
* also with Patavina Technologies s.r.l., Italy

***Abstract*—Currently one of the most established protocols for machine to machine (M2M) communications is LoRaWAN, designed to provide low power wide area network with features specifically needed to support low-cost, mobile, secure bi-directional communication for the Internet of Things (IoT). In this context security is of pivotal importance, as IoT constitutes a pervasive network of devices highly integrated with our daily life. In this paper we examine key security issues of the procedure used in LoRaWAN to allow an end device to establish a connection with the network server. We have identified vulnerabilities in this protocol, in particular with reference to the use of a random number in the join procedure packet, meant to prevent replay attacks. We first discuss the options that a network server has when detecting a replay attack and then we examine a) the possibility that a legitimate receiver is considered an attacker because of the random number generation issues and b) the possibility for an attacker to exploit this protocol to generate a denial of service (DoS). A wide set of experiments has been conducted using a widely used LoRaWAN chip showing the vulnerabilities of the protocol.**

***Index Terms*—LoRa; LoRaWAN; Internet of things (IoT); Authentication; Random Number Generator; Security.**

I. 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 has been identified as a target for the future 5G communication system, while several proposals already exist to adapt the 4G technologies to IoT. Nevertheless the IoT paradigm may present some very specific features that require new network deployments, such as low power wide area networks (LPWAN). Among the various LPWAN technologies we focus in this paper on LoRaWAN.

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 [5]. 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 the network.

In this paper we focus on the procedure protocol by which a node join a LoRa network. The join protocol is also run typically once per day to confirm the presence of nodes in the network. In order 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 [3, Ch.6]. Here we focus on the OTAA procedure. The procedure provides an exchange of two messages between the end device and the network server (*join request* and *join accept* messages). In order to avoid replay attacks of the join request, a random number (denoted DevNonce) is included in the transmitted packet, and the network server controls if the DevNonce has been already used by the same device.

Various security problems are related to this procedure and are addressed in this paper. First, it is possible that the random generator of the device generates the same DevNonce: we assess the probability of this event and the possible actions that can be taken by the network server. Second, it is not trivial to have a truly random generator for DevNonce, especially for low-cost devices. In the SX1272 transceiver [6] the random bit sequence is obtained collecting the least significant bit of the received signal strength indicator (RSSI). We analyze key parameters to establish the quality of this random number generator, indicating specific attacks that can be performed to reduce the randomness of the obtained number, down to the generation of a constant number. The success of this attack would lead to a denial of service (DoS) since one or more nodes would not be able to access the network. A wide set of experiments has been conducted using the WiMOD SK-iM880A [18] device, showing the vulnerabilities of the protocol.

The rest of the paper is organized as follows. We first analyze the join procedure and suggest two polices of the network server in case of a replay attack in Section II. Section III analyzes the security issues of the random generator, which is used for the generation of the DevNonce and outline possible attacks by a node intending to generate a DoS. A wide set of experimental results with the SX1272 transceiver are then presented and discussed in Section IV, before conclusions are outlined in Section V.

978-1-5090-5908-9/17/$31.00 ©2017 IEEE

II. Join Procedure Analysis

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. It consists of the end device identifier (DevEUI), a global end device identifier in IEEE EUI64 address space; the application identifier (AppEUI), a global application identifier in the IEEE EUI64 address space and a nonce of 16 bits (DevNonce). The join-request message is not encrypted. The network server responds 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.

Upon reception of a join request, the network server checks if the DevNonce has been already used by the device, comparing the received sequence with the last N_D sequences, where N_D is a system parameter to be chosen. If a match is found, two policies are typically implemented:

a) the network server drops the requests and waits for further requests from the same end device with a valid DevNonce or
b) the network server permanently excludes the end device from the network.

Although the protocol provides option a), some vendors have implemented option b), and therefore we analyze its consequences in this section.

For the security analysis, let f_J denote the number of valid join procedures per day per each end device 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

$$T_r = \frac{N_D}{f_J}. \tag{1}$$

A larger T_r clearly provides a higher security. To this end, assuming that f_J is an unchangeable quantity, it is advisable to have a large number of stored DevNonces. Since DevNonce is a 16-bits integer, the maximum value of N_D is 2^{16} and $T_r \leq 2^{16}/f_J$.

We now consider the effects of the two policies implemented by the network server.

Case a): If the network server only drops the requests with already used DevNonce, we are interested in the probability of generating a stored DevNonce given N_D. Supposing that the value of DevNonce has uniform probability in the alphabet $[1, \ldots, N = 2^{16}]$, and denoting S the set of stored DevNonces, with $|S| = N_D$, the probability of generating a stored DevNonce given N_D is

$$\mathbb{P}[d_K \in S] = \frac{N_D}{N}\,, \tag{2}$$

where d_k is the devNonce generated at the k^{th} join procedure. We can observe that this probability is higher if N_D is larger. So there exists a trade-off between the probability in (2) and T_r in (1).

Case b): If the network server implements the policy in which the end device is excluded if a join request with a stored DevNonce arrived, it is important to evaluate the probability to be excluded within a given time. The probability of generating K different values of devNonce is

$$\begin{aligned}
&\mathbb{P}[K \text{ different devNonce}] = \mathbb{P}[d_2 \notin \{d_1\}]\cdot \\
&\cdot \mathbb{P}[d_3 \notin \{d_1, d_2\}] \cdot \ldots \cdot \mathbb{P}[d_K \notin \{d_1, \ldots, d_{K-1}\}] = \\
&= \frac{N-1}{N} \cdot \frac{N-2}{N} \cdot \ldots \cdot \frac{N-K+1}{N} = \prod_{i=1}^{K-1} \frac{N-i}{N} = \\
&= \frac{D_{n,k}(N,K)}{N^k} = \frac{N!}{(N-K)!N^K} = \mathcal{D}(K)
\end{aligned} \tag{3}$$

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

$$\begin{aligned}
&\mathbb{P}[d_K \in \{d_1, \ldots, d_{K-1}\} \bigcap d_1 \neq d_2 \neq \ldots \neq d_{K-1}] = \\
&= \mathbb{P}[K-1 \text{ different DevNonces}]\mathbb{P}[d_K \in \{d_1, \ldots, d_{K-1}\}] = \\
&= \mathcal{D}(K-1) \cdot \frac{K-1}{N} = \mathcal{E}(K)
\end{aligned} \tag{4}$$

The probability in (4) corresponds to the probability to be excluded at K^{th} attempt if $K \leq N_D + 1$. If we consider also the case with $K > N_D + 1$ the probability is

$$\begin{aligned}
&\mathcal{S}(K|N_D) = \mathbb{P}[\text{node is excluded at } K|N_D] = \\
&= \begin{cases} \mathcal{E}(K) & \text{if } K \leq N_D + 1 \\ \mathcal{E}(N_D+1)\left(1 - \frac{N_D}{N}\right)^{(K-1-N_D)} & \text{if } K > N_D + 1 \end{cases}
\end{aligned} \tag{5}$$

Then the probability to be excluded within T attempts given N_D is

$$P_{off} = \sum_{K \leq T} \mathcal{S}(K|N_D). \tag{6}$$

A. LoRa Performance

Let's now consider typical values of f_J. In order to guarantee a correct operation of the network, at least one valid join procedure per day is typically 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 $3650 \times 2 = 7300 = N_D^{max}$ DevNonces.

Let's firstly analyze the policy a). Let's suppose that the network server decides to store $N_D < N_D^{max}$, so by (1) $T_r < 10$ years. Upon a replay attack, the legitimate end device is disconnected and if it becomes aware of this it can perform a new (valid) join request. However at each replay a new DevNonce is used and after T_R attacks the end device is permanently disconnected from the network as no new DevNonce is available.

So it seems to be better to store all the previously used values of DevNonce for each end device. But in this case, considering $N_D = N_D^{max}$ the probability of generating a previously used DevNonce, using (2), is

$$\mathbb{P}[d_K \in S] = \frac{N_D^{max}}{2^{16}} = \frac{7300}{65536} \simeq 0.11. \quad (7)$$

So it is relatively probable to generate an already stored used DevNonce in the last days of life of an end device. However, the device can 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.

Case b) where the network server excludes nodes that generate invalid join request is instead more critical. First of all we consider the plot in Fig. 1 where we reported the probability P_{off} for different values of N_D. We can notice that the larger is N_D the higher is the probability to be excluded 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 (4), we have

$$\rho = E\left[\mathcal{E}(K)\right] = \sum_{k=1}^{N} k\mathcal{E}(k) \simeq 319.5 \quad (8)$$

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

$$\mathbb{P}[\text{node is excluded within a year}] \simeq 0.98.$$

Another DoS attack provides that the malicious node registers all the join requests of end devices around it and sends them to the network server after a period. Then the network server responds excluding all the nodes for which a replayed join request has been sent. Considering that the attacker 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 permanently excluding the node from the network. Moreover the value of N_D must be higher than the estimated number of join procedures in the life of an end device.

B. DevNonce As Sequential Number

Considering the problems highlighted in the previous section we propose to use as DevNonce a sequential number. This will guarantee that no previously used values are re-used. In this case we must be sure that both the network server and the end device know the last used DevNonce value, which is ensured by the feedback of the network server, the join accept packet. If this packet is not received upon a legitimate request, the end-user must re-use the same DevNonce value since it has not reached the network server.

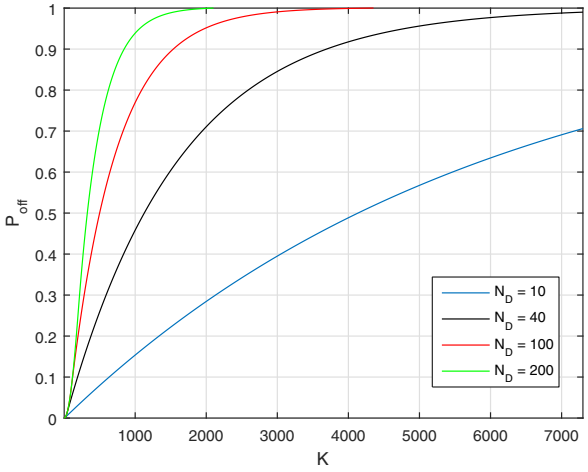

Figure 1. Probability to be excluded within K attempts.

III. Random Number Generation

The random number generator should generate independent and identically distributed random numbers X, with a uniform distribution in a set $\{x_1, \ldots, x_k\}$. This corresponds to have maximum entropy, defined as

$$H(X) = E[-\log_2 P(X)] = \sum_{i=1}^{k} -P(x_i)\log_2 P(x_i), \quad (9)$$

where $P(X)$ is the probability mass function of X. When the random number generator is meant to generate numbers that are secret to an attacker, the entropy reveals the average effort that an attacker has to apply to know the generated number. 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 finding the secret number. If the probability of different values is unequal, then a lower entropy is obtained, and fewer attempts will be required on average. While the above analysis is correct on average, it can be misleading in some cases for cryptographic analysis where the work factor for an adversary matters. For example, assume that 128 bits are generated, but that half of the time all bits are zeros and a random selection from the remaining $2^{128} - 1$ values the rest of the time. The entropy of this random number is 64 bits, but the all-zero key has half probability of being the secret key. Thus, for cryptographic purposes, it is also useful to look at the min-entropy, defined as

$$H_{\min}(X) = -\log_2 \max_i P(x_i). \quad (10)$$

The pseudo-random generator described above provides $H_{\min}(X) = 1$ bit, which corresponds to the possibility of

obtaining the all-zero secret key with probability of one half.

The National Institute of Standards and Technology (NIST) 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. In [15] an entropy source model is described, including a noise source, an optional conditioning component and a health testing component.

Noise source: The noise source is the root for the entropy source. This component contains a non-deterministic, entropy-providing activity that is ultimately responsible for the uncertainty associated with the bit strings.

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 continues 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.

A. RBG of SX1272

We focus in this paper on the procedure used in the SX1272 modem by Semtech. In this chip [6, Ch.4] an N-bits random number is obtained by performing N read operation of the least significant bit (LSB) of the register RegRssiWideband (address 0x2c). The value from RegRssiWideband is derived from a wideband (4 MHz) signal strength at the receiver input every 1 ms. It is assumed that the LSB constantly and randomly changes due to the noise and radio channel behavior (reflections, fading, shadowing, interference).

With respect to the NIST model, the RSSI is the noise source. The output of the digitization process corresponds to the value written in the RegRssiWideband register that goes from s to $s + 127$ dBm, where s is the sensitivity level, and with step of 0.5 dB. Finally the selection of the LSB in the register corresponds to the post-processing or to the conditional component, aiming at reducing biases of the entropy sources.

We note that in this procedure no health test is provided, thus making the system more vulnerable to attacks.

B. Attacks on RBG

We now consider two attacks to reduce the randomness of the random bit generator:

a) We induce saturation of the receiver by high power jamming;
b) We jam the receiver with a fixed power (not inducing saturation) in order to force a given RSSI value.

In these two cases the LoRa end device is susceptible to a DoS attack. Indeed, the value of DevNonce will be the same whenever a new join procedure is triggered by the end device and the network server will discard the join-request message.

In case a) the RegRssiWideband register will store the maximum value (all ones in binary format) since the receiver power is higher than maximum value and remains constant in time. Indeed, if the saturation value is significantly exceeded, the probability that noise brings the power below saturation is very low. In order to saturate the receiver we must transmit a high power signal. However, it is not easy to constantly saturate the receiver due to the presence of the automatic gain control (AGC) in the end device, which attenuates the received signal even for high power values (obtained by placing two LoRa devices very close to each other). However we have also to consider the response time of the AGC: if a device alternatively transmits with high and low power, the saturation of the RSSI value is possible for few instants. We will discuss this option in more details in the next Section.

In case b), since the RegRssiWideband value is quantized, it is possible that with a high and constant received power, the value written in the register changes negligibly, becoming de facto constant in time (if the received power is constant). Note that we do not need very high power in this case, as it was the saturation attack.

IV. Experimental Results

We have performed a set of experiments with the WiMOD SK-iM880A device, investigating the performance of the DevNonce generator a) without attacks, and b) in the presence of a jammer.

A. Performance Without A Jammer

We first have assessed the performance in the absence of attacks. We are interested in random bit generator and in the accuracy of the DevNonce random generator. For the random bit generator we have computed from experiments the sampling probabilities of generating zeros and ones, denoted $P[0]$ and $P[1]$ respectively, and the probability to generate sequences of bits. In order to assess the performance of the DevNonce random generator we estimate its *entropy* and *min-entropy*, defined in (9) and (10).

For the random bit generator, Table I shows the sampling probabilities obtained by experiments, where we focused on sequences of one or two bits. We observe that the probabilities of generating a 1's and 0's are slightly unbalanced, and we have observed that the displacement is due to the receiver saturation. On the other hand generated bits are independent, at least as long as we observe two consecutive bit generations since

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

are approximately equal to the sampling probabilities.

For the DevNonce generation performance, we have obtained $H(x) = 15.90$ and $H_{\min}(X) = 14.20$. Note that for an ideal random bit generator (with equal probabilities for the two values) we should have $H(x) = H_{\min} = 16$. With the

Table I
RANDOM BIT GENERATOR WITHOUT JAMMER.

Parameter	Value	Parameter	Value
P[0]	0.468	P[00]	0.219
P[1]	0.532	P[01]	0.249
P[10]	0.249	P[11]	0.283

Table II
RANDOM BIT GENERATOR WITH JAMMER AT 1 M.

Parameter	1st exp	2nd exp
Message	random	ones
P[0]	0.554	0.446
P[1]	0.446	0.554
P[00]	0.309	0.203
P[01]	0.245	0.243
P[10]	0.245	0.243
P[11]	0.201	0.311

practical random generator of the WiMOD SK-iM880A device on average, every number is repeated every $2^{15.9} \simeq 61147$ times. From the min-entropy we obtain that the most probable value appears on average every $2^{14.2} \simeq 18820$ generations instead of $2^{16} = 65536$. Using the results of the previous section, on average the most probable value is regenerated after $\frac{18820}{7300} = 2.6$ lives.

Lastly, we have estimated the value of ρ, whose theoretical value when using an ideal random bit generator has been derived in (8). From the experiments we obtained $\rho = 320.6$ which is very close to the value provided by (8), thus proving the appropriateness of the random generator in the absence of an attacker.

B. Performance With a Jammer

We now analyze the performance when a jammer is active, where the jammer is another WiMOD SK-iM880A, transmitting in LoRa mode with power of 14 dBm. Due to the jammer transmissions, we expect for example that the legitimate receive saturates more often, thus increasing the unbalance between $P[0]$ and $P[1]$.

We first position the jammer at a distance of 1 m from the legitimate receiver. Two experiments have been conducted: in the first a random message is transmitted by the jammer, in the second a string of ones is transmitted by the jammer (thus yielding a constant power). Table II shows the probabilities of bit generation. 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.

As anticipated in Section III.B we have observed is that, at the beginning and at the end of a jammer transmission saturation of the legitimate receiver occurs. This is due to the AGC of the receiver that attempts to adjust the attenuation of the received signal in order to exploit at best the dynamic range of the analog to digital converter (ADC). However, there is a delay in the adaptation, so when the jammer starts its transmission the AGC increases the attenuation but the first RSSI measurements still show saturations. The same occurs at the end of a jammer transmission, where the sudden drop of received power is not immediately compensated by the AGC, therefore we still have some saturation. Since saturations are associated to fixed least significant bits, by alternating on and off phases of the jammer we can induce saturations at the legitimate receiver, therefore determining the DevNonce.

Table III
DEVNONCE GENERATOR WITH JAMMER AT 1 M.

Parameter	1st exp	2nd exp
Message	random	ones
$H(X)$	15.87	15.95
$H_{\min}(X)$	13.87	14.64
ρ	304.8	304.3

Table IV
RANDOM BIT GENERATOR WITH JAMMER AT 0.35 M.

Parameter	1st exp.	2nd exp.
Modulation	LoRa	LoRa
Message	random	ones
P[0]	0.494	0.569
P[1]	0.506	0.431

We now examine the generation of DevNonce, whose performance is reported in Table III. We observe that entropy and min-entropy are similar to those obtained without jammer. In particular the best values are achieved when a string of ones is transmitted. Concerning the last parameter in Tab. III, 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 the first regeneration, with respect of that evaluated theoretically, even if the reduction is not so relevant (around 5%).

However, at a distance of 1 m, other phenomena such as channel fluctuations that are different from thermal noise seem to be still relevant. For this reason we set the jammer closer to the receiver, in order to decrease the contribution of this phenomena. We now position the jammer at a distance of $\lambda \simeq 0.35$ m. Also in this case two experiments have been conducted: in the first experiment we transmitted a random message; in the second we transmitted a message containing a string of ones.

With reference to Table IV, we observe more different results for the two experiments. 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 that actually appears as random noise at the receiver.

Furthermore, as seen in previous sections, since the variance of the message is higher, the AGC works with lower efficiency and we obtain a relatively high value of RSSI. Moreover if the

variance of the message is small the most probable value of RSSI is quite similar to that obtained at a distance of 1 m in the same setting. This confirms the hypothesis that AGC attempts to set the receiver in the same condition (of variance and mean), independently on the received power; however if the variance of message is relevant the AGC cannot properly follow the input dynamics.

In the second experiment, instead, we obtain an unbalanced probability of generation of bits similar to the case of jammer at 1 m.

Table V
DEVNONCE GENERATOR WITH JAMMER AT 0.35 M.

Parameter	1st exp.	2nd exp.
Message	random	ones
$H(X)$	15.94	15.84 7
$H_{\min}(X)$	14.10	12.66
ρ	293.8	271.1

We now analyze the performance of the DevNonce generator, reported in Table V. In the first experiment, with a random message transmitted by the jammer, the results are similar to those with jammer at 1 m and without jammer. In the second experiment, 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.

Finally, with a jammer at distance λ, the reduction of the first regeneration value is more relevant in the first and second experiment. The decrease is up to 18% in the second experiment (that has the lowest min-entropy).

V. CONCLUSIONS

In this paper we have examined the security of the join procedure of the LoRa protocol. We have highlighted various issues, in particular we have examined the probability of a DoS even without the presence of an attacker, due to the regeneration of an already used DevNonce by the end device. We have considered the options available at the network node in case a replay attack is detected and their consequences. We then examined the random number generator used in the SX1272 modem and we have proposed attack strategies by a malicious device aiming at generating DoS. Experimental results show the inaccuracies of the current implementation of the random number generator and potential vulnerabilities to DoS attacks. Further investigations are needed to assess other attacks such as the alternation of on and off phases of the attackers to yield saturation of the legitimate receiver, thus yielding to a fixed DevNonce number.

ACKNOWLEDGMENTS

We would like to thank Ivano Calabrese from Patavina Techologies for his support.

REFERENCES

[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