Evaluation of Side-Channel Key-Recovery Attacks on LoRaWAN End-Device

Kazuhide Fukushima[1](✉), Damien Marion[2], Yuto Nakano[1], Adrien Facon[2], Shinsaku Kiyomoto[1], and Sylvain Guilley[2]

[1] KDDI Research, Inc., 2–1–15 Ohara, Fujimino, Saitama 356–8502, Japan
ka-fukushima@kddilabs.jp

[2] Secure-IC S.A.S., 80 Avenue des Buttes de Coësmes Rennes, 35700 Cesson-Sévigné, France

Abstract. IoT devices have come into widespread use. The rapid growth of the IoT market is expected in the field of automobiles and transportation, medical and health care, and industry. Data protection and integrity are critical for IoT-based services in order to maintain the security and privacy of them. Low-power wide-area (LPWA) is a wireless communication technology designed for IoT applications and end-devices requiring low cost, long battery life, wide-area coverage, and high system capacity. LoRaWAN is an open standard for LPWA and achieves data protection and integrity by using encryption and message integrity code (MIC). Many studies have pointed out security issues, and attacks against LPWA protocols and have proposed solutions to improve security against such attacks. However, side-channel analysis techniques can directly recover secret information from a device. In this paper, we evaluate the applicability of a side-channel analysis to a real LoRaWAN end-device. Our experiments attempt to recover AES-128 keys to encrypt frame payload and calculate the message integrity code (MIC) for the encrypted payload based on a correlation power analysis, which is a type of side-channel analysis. The 260 electromagnetic(EM)-leakage traces entirely recover the 16-byte key for the frame payload encryption, and the 140 EM-leakage traces recover the 12 bytes of the 16-byte key for MIC generation. Furthermore, we show that our key recovery attack is applicable in real LoRaWAN protocols. Our attack can entirely recover the root key AppKey in LoRaWAN v1.0 and a root key NwkKey in LoRaWAN v1.1.

Keywords: Internet of things (IoT) · Low-power wide-area (lpwa) · Lorawan side-channel analysis · Correlation power analysis (cpa) electromagnetic(EM)-leakage · AES

1 Introduction

Rapid growth of the IoT market is expected in the areas of automobiles and transportation, where the use of connected-vehicles is expanding, the medical

A preliminary version of this paper [7] appeared in Proceedings of the 5th International Conference on Information Systems Security and Privacy (ICISSP 2019).

field, where we see growth in the use of digital devices for healthcare, and in industry (including factories, infrastructure, and logistics), where we are witnessing the expansion of smart factories and smart cities. IHS Markit [10] predicts that the number of connected IoT devices worldwide will increase by 12% on average annually.

Low-power wide-area (LPWA) is a term used to describe wireless communication technologies for IoT applications. These technologies are characterized by low cost, long battery life (or low power consumption), wide-area coverage, and high system capacity. LPWA technologies can be roughly categorized into the licensed and unlicensed spectrum. LoRaWAN [17] and Sigfox [21] are typical examples of protocols that run in the unlicensed spectrum, and a license is not needed to build a network and provide services. LoRaWAN and LoRa are open standards designed by the LoRa Alliance. LoRaWAN defines the communication protocol and system architecture in the medium access control (MAC) layer for the network, while LoRa defines the physical layer or wireless modulation that enables wide-area coverage. Everyone can build LoRaWAN network can be built by purchasing equipment similar to a wireless LAN. Conversely, only one company in each country can build a Sigfox network according to the policy of the Sigfox company. LTE-M [9] and NB-IoT [1] operate over the licensed spectrum, and only mobile operators build a network and provide services. Their advantage is that existing LTE base stations can be used to build a new LPWA network.

Data protection and integrity are critical for IoT-based services. For example, user privacy may be compromised by location information and activity information acquired from a wearable device. As another motivating example, an air conditioner can be manipulated maliciously by modifying the value of the temperature sensor, which can lead to panic in crowded places. In many cases, LPWA technologies achieve data protection and integrity by using encryption and message integrity code (MIC). However, many attacks against the vulnerabilities of LPWA protocols have been proposed, and some of them are potential threats as they can extract the secret keys. Furthermore, side-channel analysis technologies exist that have the capacity to recover secret information from devices by using side-channel information, including timing information, power consumption, electromagnetic leaks, sound, and heat.

Our Contributions. We evaluate the applicability of a side-channel analysis technique to a real LoRaWAN end-device. Our experiments attempt to recover the AES-128 keys from the EM-leakage traces produced by the AES-128 encryption algorithm payload encryption process and MIC generation process for data transmission on a real LPWA end-device. The 350 electromagnetic(EM)-leakage traces of payload encryption process can recover the entire AES key. The required number of EM-traces can be reduced to 260 using a band-pass filtering technique. To the best of our knowledge, this is the first paper that describes a key recovery attack from a real LPWA device with less than 300 EM-leakage traces. The 140 EM-leakage traces of the MIC generation process can recover 12-byte of the AES key, except for the first four bytes. The remaining four bytes can be obtained by brute-force guessing with 2^{32} computational complexity to recover the entire key. Furthermore, we show that our key recovery attack is applicable in real LoRaWAN protocols. Our attack can entirely recover the root key AppKey in LoRaWAN v1.0 and a root key NwkKey in LoRaWAN v1.1.

A preliminary version of this paper [7] appeared in the Proceedings of the 5th International Conference on Information Systems Security and Privacy (ICISSP 2019). This full-version provides a discussion regarding the applicability to real LoRaWAN v1.0 and v1.1 while the previous paper demonstrated potential threats of side-channel key-recovery attacks against payload encryption and MIC generation processes. Furthermore, we add some omitted data in the preliminary paper, including the correlation between the EM-leakage and Hamming weight for all the bytes of the recovered key.

2 Related Work

State-of-art studies have pointed out security issues, and attacks against LPWA protocols and have proposed solutions to improve security against such attacks. Most of them target the LoRaWAN protocol since it is an open standard, and the specification is publicly available. Girard [8] pointed out the issues in key provisioning for LoRaWAN end-devices. Zulian [25] and Tomasin et al. [23] demonstrated the possibility of a replay attack against the join procedure in LoRaWAN. The replay attack is due to the limitation in the variety of the DevNonce generated by an end-device, and theoretically and experimentally showed that random number generators in a real end-device are not secure. Na et al. [20] argued that LoRaWAN was vulnerable to a similar replay attack and described countermeasures, and Lee et al. [15] proposed a bit-flipping attack against an encrypted frame payload using AES-CTR and a countermeasure. Yang et al. [24] discovered several vulnerabilities of LoRaWAN and demonstrated five types of attacks: 1) replay attack leads to a selective DoS attack, 2) plaintext recovery attack, 3) malicious message modification, 4) falsification of delivery reports, and 5) battery exhaustion attack. A selective jamming attack against the LoRa physical layer and its countermeasure is proposed by Aras et al. [2]. Butun et al. [4] demonstrated five types of attacks: 1) RF jamming attack, 2) replay attack, 3) Beacon (Class B) synchronization attack, 4) network traffic analysis and 5) man-in-the-middle (MITM) attack against the latest version: LoRaWAN specification v1.1 released on Oct 11, 2017.

Side-channel analysis can recover secrets of a device based on side-channel information such as sound, heat, timing information, power consumption, and electromagnetic-leakage. Some existing studies target IoT end-devices or resource-constrained devices. Kocher et al. [13] were the first to propose a side-channel attack. They leveraged a device's power consumption on a device and demonstrated that a DES key can be recovered. Their attack contains a simple power analysis (SPA), differential power analysis (DPA), and higher-order DPA (HO-DPA). Messerges et al. [18] theoretically derived the signal-to-noise ratio (SNR) in a DPA attack against DES proposed by Kocher et al., and improved the DPA to d-bit DPA by focusing on multiple bits in the S-Box of DES.

A DPA attack against the scalar multiplication on an elliptic curve-based cryptosystem (ECC) was proposed by Joye and Tymen [12]. Itoh et al. [11] proposed a DPA attack focusing on the register address of an ECC. Brier et al. [3] were the first to study a correlation power analysis (CPA) based on a Hamming distance leakage model. The CPA utilizes the correlation between the leakage model of a sensitive value and its power consumption or electromagnetic(EM)-leakage. The Hamming weight leakage model, proposed by Kocher et al. [13] and Messerges et al. [18], assumes that leakage through the side-channel depends on the number of bits set in the data. The leakage value T_{HW} of data X can be formulated as

$$T_{\mathsf{HW}} = a\mathsf{HW}(X) + c + \sigma$$

where a is a coefficient, $\mathsf{HW}(\cdot)$ is the Hamming weight function, c is a constant leakage, and σ is noise. The Hamming distance leakage model assumes that leakage depends on the number of bits switching from one state to another. The leakage for a bit switching from 0 to 1 and from 1 to 0 are assumed to be same. The leakage value T_{HD} in the case where data X change to X' can be formulated as

$$T_{\mathsf{HD}} = a\mathsf{HW}(X \oplus X') + c + \sigma.$$

Komano et al. [14] proposed a build-in determined sub-key CPA (BS-CPA) that finds a new sub-key by using the previously determined sub-keys recursively and demonstrated that it can recover a DES key with fewer power traces than the original CPA. Clavier et al. [5] applied a CPA to first-order protected AES implementations and showed that the CPA requires fewer power traces than classical second-order DPA. Dinu and Kizhvatov [6] showed that a DPA can recover a partial AES-CCM key on a wireless microcontroller. Tawalbeh and Somani [22] evaluated the security of AES, ECC, and RSA against timing and fault side-channel attacks and showed countermeasures for IoT implementation. A side-channel evaluation platform for IoT end-devices is proposed by Moukarzel et al. [19].

3 Key Recovery Attack

We propose a key recovery attack based on correlation power analysis, a type of side-channel analysis. Our attack is applicable to general LoRaWAN end-devices. The goal and assumptions are described, and then the details of the attack are explained.

3.1 Goal

LoRaWAN protocol uses Advanced Encryption System (AES), a symmetric encryption algorithm to achieve the security and integrity of transmitted data. Data protection is ensured using AES-CTR, and message integrity is guaranteed by the computing of a message integrity code (MIC) using AES-CMAC. Our key recovery attack thus targets AES-128 keys for payload encryption and

MIC generation stored in an end-device. An attacker can decrypt or forge all messages and commands transmitted between the server and end-devices, or connect malicious end-devices to the LoRaWAN network by abusing these keys.

3.2 Assumptions

An attacker as a security evaluator assumed to be able to access plaintext. The attacker does not have to take control of the plaintext and the corresponding ciphertext. This condition can be met if the attacker knows the data format and the data itself. For example, an end-device sends current temperatures periodically, and the attacker can guess the plaintext using a separate thermometer. Another way to meet the assumption is to access an API for data transmission on an end-device. Some LoRaWAN libraries provide APIs for data transmission that takes plaintext messages or commands as input. Our key recovery attack is based on correlation power analysis and requires multiple pairs of plaintext and ciphertext. This attack is not applicable to fixed messages such as prefixed values in a protocol header since the Pearson correlation coefficient cannot be calculated. However, we can recover the keys and all the messages from a small number of partial plaintext. In our experiments, we modify the source code of a LoRaWAN end-device to set a trigger signal at the first round of AES-128. However, modification of the source code is not essential if different EM-leakage traces can be appropriately aligned along the time axis.

Our proposed attack is focused on the first round of AES-128, using the knowledge of the plaintext and guessing each byte of the AES-128 key independently. Guessing each byte of the first round key allows each byte of the output of the S-Box to be recovered independently at the first round. The first round of AES-128 consists of four operations: AddRoundKey, SubBytes, ShiftRows and MixColumns. Figure 1 shows the detailed processes of the first round of AES-128.

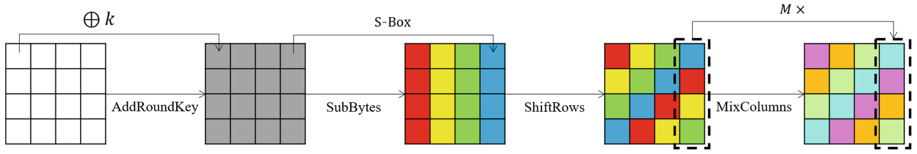


Fig. 1. First round of AES-128.

3.3 Key Recovery Attack

We now describe the key recovery attacks in detail. Our key recovery attack consists of a leakage identification phase and key recovery phase.

Leakage Identification. The first phase of the attack is to identify the EM-leakage traces produced by the AES-128 encryption algorithm. The EM-leakage of hundred executions with the same key and plaintext has been averaged. This process permits an increase in the signal-to-noise ratio (SNR) defined as

$$\mathrm{SNR} = \frac{P_{\text{AES-128}}}{P_{\text{Noise}}},$$

where $P_{\text{AES-128}}$ and P_{Noise} are the power of AES-128 leakage and the noise, respectively. The noise P_{Noise} can be considered to follow a Gaussian distribution $N(\mu, \sigma^2)$ that explains the increase in the SNR by averaging. Figure 2 displays the result of this recording. This graph permits the ten rounds of AES-128 to be identified, and we can delimit each round. This delimitation revealed a repetition of four events in each round (identified by four peaks) and corresponds to AddRoundKey, SubBytes, ShiftRows and MixColumns of AES-128. The x-axis represents time (i.e., the number of samples), and the y-axis represents the electromagnetic range in volts.

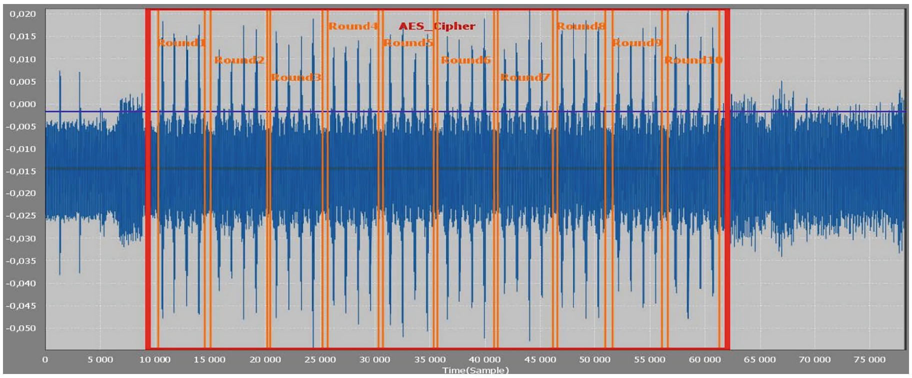


Fig. 2. EM-leakage of hundred AES-128 executions [7].

Key Recovery. The second phase of the attack is to recover the AES-key based on analysis of the EM-leakage traces. Our key recovery attack uses correlation power analysis [3] with the Hamming weight leakage model and focuses on the output of the SubBytes operation in the first round. The Hamming weight leakage model is justified by the fact that it is a software implementation. The following steps describe an algorithm to compute the correlation between the EM-leakage and Hamming weight:

1. Record the EM-leakage traces produced by AES-128 encryption algorithm AES-128$(*, P_i)$ and store them to $X^{d,i}$. Note that $*$ is the unknown AES-128 key, P_i is the plaintext in i-th trace, and $X^{d,i}$ $(0 \leq d < D$ and $0 \leq i < Q)$ is the d-th sample in the i-th trace out of Q traces of D samples.

2. Compute the guessed distributions (one by key byte):

$$Y_{i,k}[b] = \mathsf{HW}(\mathsf{SubBytes}(P_i[b] \oplus k))$$

   for $0 \leq k < 256$, $0 \leq i < Q$, and $0 \leq b < 16$ where $P_i[b]$ is the b-th byte of P_i, k is the guessed value of the key byte. $Y_{i,k}[b]$ is a (16×256)-guessed distributions of Q elements.
3. Compare $X^{d,i}$ and all the (16×256)-guessed distribution $Y_{i,k}[b]$ using the Pearson correlation coefficient [7]:

$$\begin{aligned} &r(k,d)[b] \\ &= \rho(X^d, Y_k[b]) = \frac{\mathrm{Cov}(X^d, Y_k[b])}{\sqrt{\mathrm{Var}(X^d)\,\mathrm{Var}(Y_k[b])}} \\ &= \frac{\sum_{i=0}^{Q-1}(X^{d,i} - \bar{X}^d)(Y_{i,k}[b] - \bar{Y}_k[b])}{\sqrt{\sum_{i=0}^{Q-1}(X^{d,i} - \bar{X}^d)^2 \sum_{i=0}^{Q-1}(Y_{i,k}[b] - \bar{Y}_k[b])^2}}, \end{aligned} \tag{1}$$

Algorithm 1. Key Recovery Attack [7].

```
    Input: Plaintext $P_i$ $(0 \leq i < Q)$
    Output: Recovered key $k^\star$
1   for $i \leftarrow 0$ to $Q-1$ do
2       for $d \leftarrow 0$ to $D-1$ do
3           $X^{d,i} \leftarrow$ EM-leakage of AES-128$(*, P_i)$;
4       end
5   end
6   for $i \leftarrow 0$ to $Q-1$ do
7       for $b \leftarrow 0$ to $16-1$ do
8           for $k \leftarrow 0$ to $256-1$ do
9               $Y_{i,k}[b] \leftarrow \mathsf{HW}(\mathsf{SubBytes}(P_i[b] \oplus k))$;
10          end
11      end
12  end
13  for $d \leftarrow 0$ to $D-1$ do
14      for $b \leftarrow 0$ to $16-1$ do
15          for $k \leftarrow 0$ to $256-1$ do
16              $r_{k,d}[b] \leftarrow \rho(Y_k[b], X^d)$;
17          end
18      end
19  end
20  for $B \leftarrow 0$ to $16-1$ do
21      $k^\star[b] \leftarrow \underset{0 \leq k < 256}{\arg\max}\{\max_{0 \leq d < D}\{r_{k,d}[b]\}\}$;
22  end
23  return $k^\star$;
```

where

$$\bar{X}^d = \frac{1}{Q}\sum_{i=0}^{Q-1} X^{d,i} \text{ and } \bar{Y}_k[b] = \frac{1}{Q}\sum_{i=0}^{Q-1} Y_{i,k}[b].$$

for $0 \leq d < D$, $0 \leq b < 16$, and $0 \leq k < 256$.

Algorithm 1 describes all the steps involved in our key recovery attack. The required number of traces to recover b-th byte of the key $N[b]$ is defined as the minimum q such that the recovered key byte $k^\star[b]$ using q' traces is identical to that using q traces for all $q' > q$.

4 Experimental Results

We show the result of our experiment of the key recovery attack against a real LoRaWAN end-device. Our experiment setup is sketched in Sect. 4.1 and results of a key recovery attack are shown in Sect. 4.2. Section 4.3 demonstrates a technique to reduce the number of EM-leakage traces to recover AES-128 keys.

4.1 Experiment Setup

We used a LoRa Starter Kit as a target device. This starter kit is composed of two end-devices: an end-device with a plug-and-play LoRa module and an

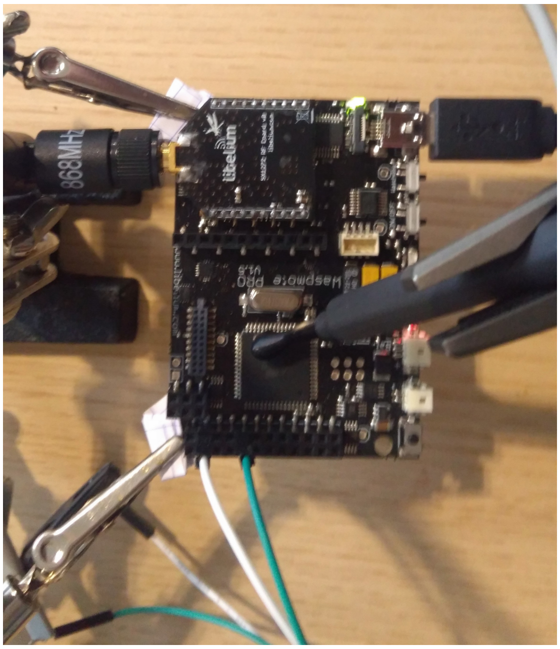


Fig. 3. EM-leakage measurement from a end-device [7].

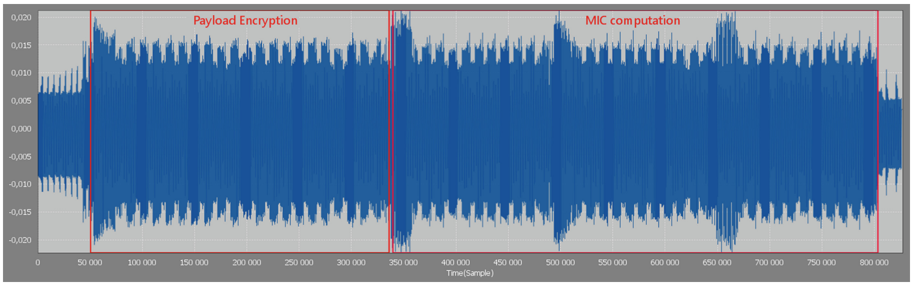


Fig. 4. Identifications of the payload encryption and MIC computation [7].

868 MHz antenna and a gateway equipped with a LoRa module and an 868 MHz antenna. We used an API supplied by the end-device for our experiment to access an implementation of AES-128. The source code of the program provided by the starter kit was modified to add a trigger signal at the first round of the AES-128.

4.2 Key Recovery

We targeted the payload encryption key k_E and MIC generation key k_M in the data transmission process in our experiment. The EM-leakage traces produced by the AES-128 encryption algorithm were recorded according to the following process.

1. The gateway generates a random plaintext P and sends it to an end-device.
2. The end-device generates a ciphertext $C = \text{AES-128}(k_E, P)$.
3. The EM-probe gets the leakage information on k_E, and the oscilloscope records the information.
4. The end-device generates $\text{MIC} = \text{AES-CMAC-128}(k_M, \text{DevAddr}||\text{FCnt}||C)$ $[0 \ldots 7]$.
5. The EM-probe gets the leakage information on k_M, and the oscilloscope records the information.
6. The end-device sends a frame including C and MIC to the gateway.
7. Goto step 1 until a sufficient number of traces is captured.

In our key recovery attack, we need to identify the distinct encryption phases in the EM-leakage traces to the first round of AES-128 for the payload encryption or MIC computation. We can find patterns in the EM-leakage traces produced by the AES-128 encryption algorithm. Figure 3 shows the measurement of the EM-leakage from a LoRaWAN end-device.

We can identify two distinct parts; the first part corresponds to the encryption of the frame payload and the second part to the MIC computation. Figure 4 shows 20 similar patterns in the signal part identified as the frame payload encryption. The frame payload is composed of 32 bytes, and AES-128 with ten rounds is executed twice; thus, 20 similar patterns appear. The same pattern can be identified three times in a row within the second part identified as the MIC computation for encrypted payload. The MIC computation for 40-byte data executes the AES-128 encryption algorithm three times. Furthermore, inside each AES-128, the same pattern could be identified ten times, which is corresponding to 10 rounds in AES-128.

Our key recovery attack is applied to the first round of AES-128. Figure 5 shows the result of the attack against the payload encryption process and plots the correlation between the EM-leakage and the Hamming weight for each byte of the recovered key $k^[0]$ with the highest correlation. These values with the highest correlation are identical to bytes of the AES-128 key, or $k^\star[b] = K[b]$ for all B. That is, our key recovery attack reveals the entire AES-128 key. In the sixteen traces of correlation, two peaks with an amplitude around 0.4 are identifiable and it suggests that the intermediate value $\{\mathsf{SubBytes}(P[b] \oplus k^\star[b])\}$ $(0 \leq b < 16)$ is manipulated at least twice. The entire AES-128 key for the frame payload encryption can be recovered with 350 electromagnetic (EM)-leakage traces. Table 1 shows the number of required EM-leakage traces $N[b]$ to recover each key-byte. On the other hand, the 140 EM-leakage traces can recover 12 bytes of the MIC calculation key; however, the four bytes from the first byte to the fourth byte of the key never converge in our key recovery algorithm. The first four bytes of the input to the first execution of AES-128 for the MIC calculation are DevAddr and constant. The variances of $\{Y_{i,k}[b]\}$ thus vanish for $0 \leq B < 4$, and we cannot obtain the Pearson correlation coefficient in Eq. (1). One way to recover the four bytes is to use brute-force guessing with 2^{32} computational complexity. Alternatively, another leakage model, such as leakage during the computation of the $\mathsf{MixColumns}$ operation, could be used.

4.3 Reduction in the Number of Required Traces

EM-leakage traces contain uncorrelated noise produced by non-cryptographic circuits, and it may increase the required number of EM-leakage traces. The targeted end-device has a frequency of 14 MHz. By computing the spectrogram of the recorded EM-leakage around this frequency, we obtain Fig. 6 where the color gradient indicates the signal amplitude as a function of time (x-axis) and frequency (y-axis). This spectrogram shows activity around 14 to 15 MHz, which corresponds to the activity of the targeted microprocessor. We can thus apply a software-based band-pass filter between 13 and 16 MHz to remove low and high-frequency noise and improve the signal-to-noise ratio. Figure 7 illustrates the effect of the de-noising process, and a raw trace is plotted in blue and the associated de-noised trace in green.

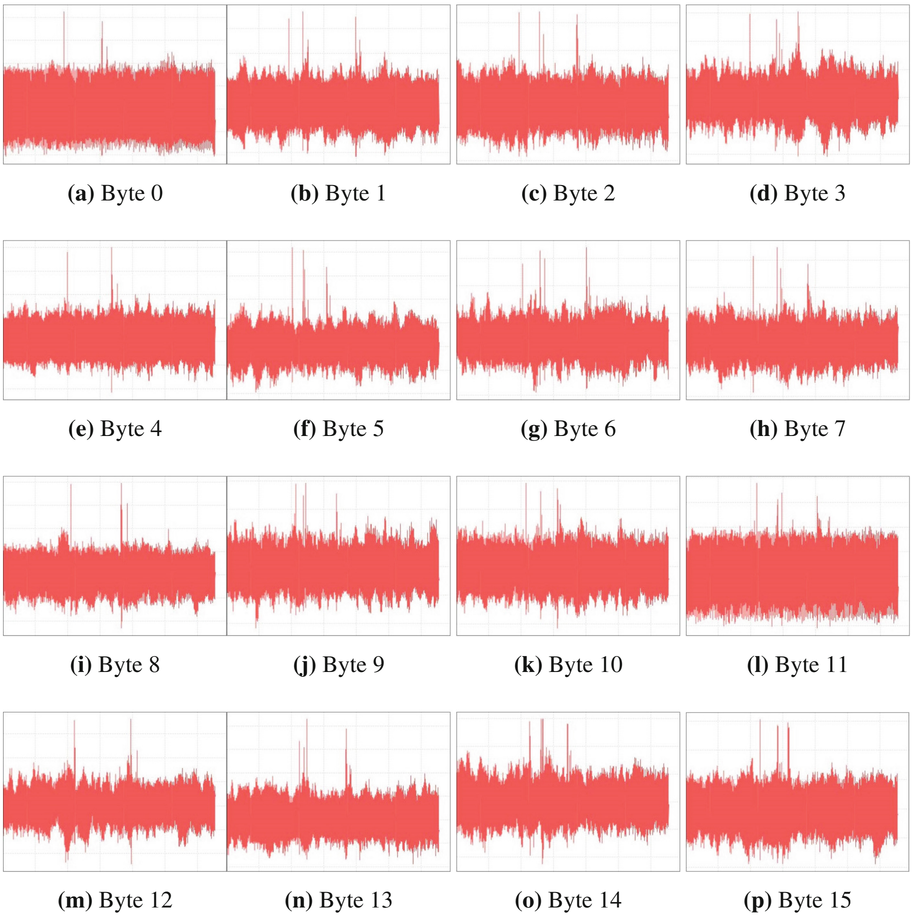


Fig. 5. Correlation between the EM-leakage and Hamming weight for each byte of recovered key $k^{\star}$.

The application of band-pass filtering on the raw traces used in Sect. 4.2 improves the efficiency of our key recovery attack. We summarized the number of required traces $N[b]$ to recover each key-byte in Table 1 to compare both results. The column "improvement" shows the difference (as a percentage) between the number of traces required to recover each key byte with the raw traces and the de-noised trace. The band-pass filtering technique reduces the number of traces required to achieve the entire key by about 26%.

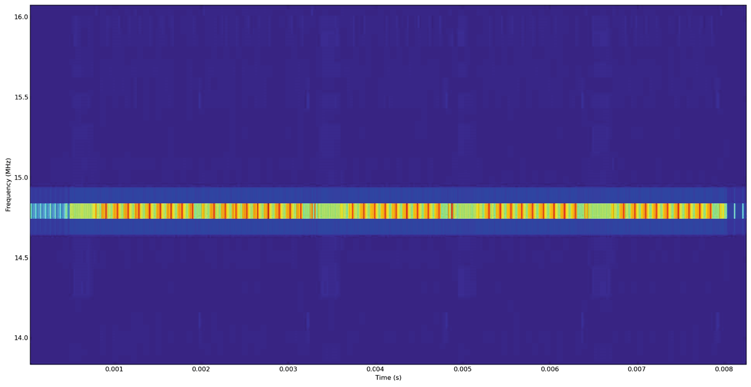


Fig. 6. Spectrum of an EM-leakage trace highlighting the activity of the microprocessor around 14–15 MHz [7]

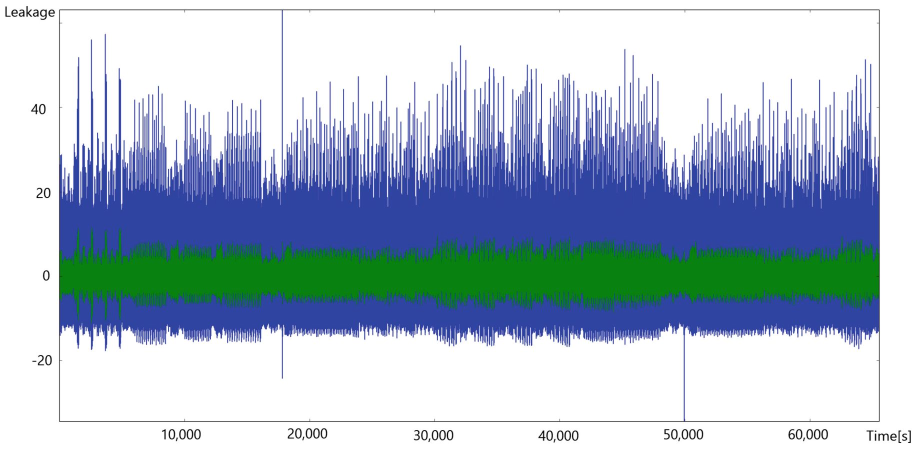


Fig. 7. Band-pass filtered trace between 13 and 16 MHz in green and raw trace in blue [7]. (Color figure online)

5 LoRaWAN Protocol

We provide an overview of the LoRaWAN protocol. The end-device activation to set AES-128 keys for an end-device is described in Sect. 5.1, and data protection and integrity in frame transmission are described in Sect. 5.2 based on LoRaWAN specification v.1.0.3 [16]. We then show the applicability of our attack to the real LoRaWAN protocol v.1.0 in Sect. 5.3 and demonstrate the difference between LoRaWan specification v.1.0 and v1.1 in Sect. 5.4.

5.1 End-Device Activation

We have to personalize and activate the end-devices to connect them to a LoRaWAN network. There are two activation methods for an end-device: over-the-air activation (OTAA) and activation by personalization (ABP).

Table 1. Required number of traces to recover the key [7].

Byte (b)	Raw	De-noised	Improvement
0	140	90	−36%
1	220	130	−41%
2	200	80	−60%
3	310	120	−61%
4	130	70	−46%
5	200	260	+30%
6	150	110	−27%
7	260	110	−58%
8	230	210	−9%
9	320	180	−44%
10	350	130	−63%
11	230	200	−13%
12	180	200	+11%
13	80	80	± 0%
14	300	90	−70%
15	210	260	+23%
Maximum	350	260	−26%

Over-the-Air Activation. In OTAA, an end-device must complete a join procedure to be able to make data exchanges with the network server. The join procedure requires the end-device to be personalized with a globally unique end-device identifier (DevEUI), an application identifier (AppEUI), and an AES-128 key (AppKey).

The join procedure in OTAA is started from an end-device by sending a join-request message. The join-request message contains AppEUI, DevEUI of the end-device, and a nonce of two bytes (DevNonce), or

$$\text{join-request msg} = \text{AppEUI}||\text{DevEUI}||\text{DevNonce}$$

AppEUI and DevEUI are a globally unique application ID of an end-device and an end-device ID in the IEEE EUI64 address space, respectively. DevNonce is a random value. The network server needs to keep the list of used DevNonce values for each end-device and ignores join requests with re-used DevNonce values to prevent replay attacks. The MIC for the join-request message is calculated as:

$$\text{MIC} = \text{AES-CMAC-128}(\text{AppKey}, \text{MHDR}||\text{join-request msg})[0\ldots3].$$

The network server responds to the join-request message with a join-accept message if the server accepts that the end-device can join an LPWA network. No response is sent to the end-device if the network server does not accept the join request. The join-accept message contains an application nonce (AppNonce) of three bytes, a network identifier (NetID), an end-device address (DevAddr), a delay between TX and RX (RxDelay), and an optional list of channel frequencies (CFList) for the network the end-device is joining, or

$$\text{join-accept msg} = \text{AppNonce}||\text{NetID}||\text{DevAddr}||\text{DLSettings}||\text{RxDelay}||\text{CFList}.$$

The MIC for the join-accept message is calculated as:

$$\text{MIC} = \text{AES-CMAC-128}(\text{AppKey}, \text{MHDR}||\text{join-accept msg})[0\ldots3].$$

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

$$\text{AES-128}^{-1}(\text{AppKey}, \text{join-accept msg}||\text{MIC}).$$

Note that AES-128 decryption is used to *encrypt* the join-accept message so that the end-device uses only AES-128 encryption to *decrypt* the join-accept message.

The network server and end-device derive the two session keys, NwkSKey and AppSKey, as follows:

$$\text{NwkSKey} = \text{AES-128}(\text{AppKey}, \texttt{0x01}||\text{AppNonce}||\text{NetID}||\text{DevNonce}||\mathsf{pad}_{16}),$$
$$\text{AppSKey} = \text{AES-128}(\text{AppKey}, \texttt{0x02}||\text{AppNonce}||\text{NetID}||\text{DevNonce}||\mathsf{pad}_{16}).$$

The function pad_{16} adds zero bytes so that the data length is a multiple of 16.

Activation by Personalization. End-devices can be activated by personalization (ABP). ABP directly associates an end-device to a LoRaWAN network without having to use the join procedure needed in OTAA.

NwkSKey, AppSKey, and DevAddr are stored in the end-device directly in ABP, while these keys are derived using the DevEUI, AppEUI, and AppKey in OTAA. The required information is preset to the end-device to connect a LoRaWAN network. Each end-device has a unique set of NwkSKey and AppSKey.

5.2 Data Transmission

Payload encryption using AES counter mode (AES-CTR-128) provides data protection of the frame payload for transmissions in the LoRaWAN protocol. AES-CMAC-128 is used to generate a four-byte message integrity code (MIC) to maintain data integrity in payload transmissions and the OTAA procedure.

Data Protection. FRMPayload is encrypted before the MIC is calculated. The encryption key K depends on the FPort of the data message: If FPort is 0, then NwkSKey is used, and if FPort is in the range of 1, 2, ..., 255, then AppSKey is used. The encryption algorithm defines a sequence of blocks A_i. A block A_i contains one-byte `0x01`, followed by four-bytes `0x00000000`, one-byte direction field (Dir), four-byte identifier (DevAddr), four-byte FCntUp or FCntDown, one-byte `0x00`, and one-byte encoded i, or

$$A_i = \texttt{0x01} \| \texttt{0x00000000} \| \mathrm{Dir} \| \mathrm{DevAddr} \| \mathrm{FCntUp\ or\ FCntDown} \\ \| \texttt{0x00} \| \mathsf{encode}(i).$$

Dir describes the direction field: 0 for uplink frames and 1 for downlink frames. The DevAddr identifies the end-device in the current network. The frame counter FCntUp, incremented by end-devices, records the number of uplinks to the network server and FCntDown, incremented by the server, records the number of downlink frames from the server. Algorithm 2 shows the procedure of the payload encryption in detail. The function len returns the byte length of the data. device must complete a join procedure

Algorithm 2. Payload Encryption in LoRaWAN Protocol [7].

```
   Input: FramePayload, Encryption key $K$
   Output: EncrypredPayload
 1 pld ← FRMPayload;
 2 $k \leftarrow \lceil \mathsf{len}(\mathrm{pld})/16 \rceil$;
 3 for $i \leftarrow 1$ to $k$ do
 4 |   $S_i \leftarrow$ AES-128$(K, A_i)$;
 5 end
 6 $S \leftarrow S_1 \| S_2 \| \ldots \| S_k$;
 7 $T \leftarrow (\mathrm{pld} \| \mathsf{pad}_{16}) \oplus S$;
 8 EncryptedPayload ← First len(pld) bytes of $T$;
   /* Data protection using AES-CTR */
 9 return EncryptedPayload;
```

Data Integrity. All LoRa messages carry a PHY payload (Payload) consisting of one-byte MAC header (MHDR), a MAC payload (MACPayload), and a four-byte MIC. The MAC payload of the data messages starts with a frame header (FHDR) followed by an optional port field (FPort) and ends with an optional frame payload field (FRMPayload). The FHDR consists of the address of the end-device (DevAddr), a frame control byte (FCtrl), a frame counter (FCnt), and frame options (FOpts) to transport MAC commands. The MIC for payload calculated on the entire message is defined as

$$\mathrm{msg} = \mathrm{MHDR} \| \mathrm{FHDR} \| \mathrm{FPort} \| \mathrm{EncryptedPayload}.$$

The block B_0 for the MIC calculation contains one-byte `0x49`, followed by four-bytes `0x00000000`, one-byte direction field (Dir), four-byte identifier (DevAddr), four-byte FCntUp or FCntDown, one-byte `0x00`, and one-byte len(msg), or

$$B_0 = \texttt{0x49} \| \texttt{0x00000000} \| \mathrm{Dir} \| \mathrm{DevAddr} \| \mathrm{FCntUp\ or\ FCntDown} \\ \| \texttt{0x00} \| \mathsf{len}(\mathrm{msg}).$$

The MIC is calculated as

$$\mathrm{MIC} = \text{AES-CMAC-128}(\mathrm{NwkSKey}, B_0 \| \mathrm{msg})[0 \ldots 3].$$

5.3 Applicability to Real LoRaWAN Protocols

The end-device executes AES encryption to calculate MIC for the join-request message, decrypts the join-accept message and verifies its MIC, derives the session keys, encrypts uplink data, calculates the MIC for the uplink data, and verifies the MIC for downlink data. The MHDR, AppEUI, and DevEUI are fixed values and DevNonce is a random value in a join-request message. We can recover up to two bytes of AppKey from the leakage during the MIC calculation for the join-request message since there are fixed values in the input data to AES. The input data to AES is $\text{AES-128}^{-1}(\mathrm{AppKey}, \text{join-accept msg} \| \mathrm{MIC})$ in the decryption process of a join-accept message. The data does not contain fixed values, and we can recover the entire AppKey from the EM-leakage traces produced by the AES-128 encryption algorithm according to our experimental results. The session keys, NwkSKey and AppSKey, can be derived from the NwkKey. The input data to AES contain some fixed values including headers and identifiers in the verification of join-accept MIC, derivation of the session keys, encryption and MIC calculation for transmitted data, and the entire AES key cannot be recovered.

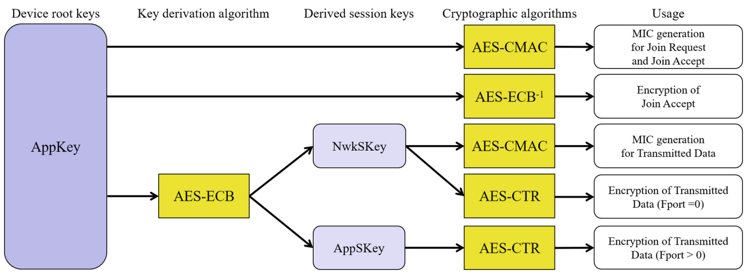

Device root keys
Key derivation algorithm
Derived session keys
Cryptographic algorithms
Usage
AppKey
AES-ECB
NwkSKey
AppSKey
AES-CMAC
AES-ECB-1
AES-CMAC
AES-CTR
AES-CTR
MIC generation for Join Request and Join Accept
Encryption of Join Accept
MIC generation for Transmitted Data
Encryption of Transmitted Data (Fport =0)
Encryption of Transmitted Data (Fport > 0)

Fig. 8. Key derivation scheme in LoRaWAN v1.0.

5.4 Difference Between LoRaWAN V1.0 and V1.1

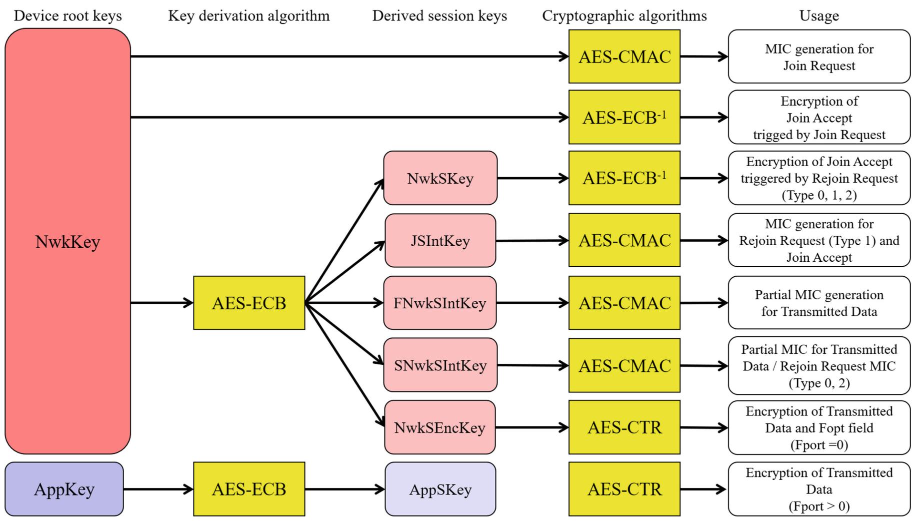
The key derivation scheme has been significantly changed between LoRaWAN v1.0 and v1.1. A new root key, NwkKey, and new session keys, NwkSEncKey, NwkSIntKey, SNwkSIntKey, and FNwkSIntKey, were added. The NwkSEncKey encrypts transmitted data where FPort is 0 similar to NwkSKey in LoRaWAN v1.0. The NwkSIntKey is used to calculate the MIC for downlink data. The SNwkSIntKey and FNwkSIntKey calculate the MIC for uplink data. The MIC is calculated as follows:

$$\begin{aligned} \text{cmacS} &= \text{AES-CMAC-128}(\text{SNwkSIntKey}, B_1\|\text{msg}), \\ \text{cmacF} &= \text{AES-CMAC-128}(\text{FNwkSIntKey}, B_0\|\text{msg}), \\ \text{MIC} &= \text{cmacS}[0,1]\|\text{cmacF}[0,1]. \end{aligned}$$

The block B_1 is defined as:

$$B_1 = \texttt{0x49}\|\text{ConfFCnt}\|\text{TxDr}\|\text{TxCh}\|\text{Dir}(=\texttt{0x00})\|\text{DevAddr}\|\text{FCntUp} \\ \|\texttt{0x00}\|\mathsf{len}(\text{msg}),$$

where the key derivation schemes of LoRaWAN v1.0 and v1.1 are given in Fig. 8 and Fig. 9, respectively.



Device root keys
Key derivation algorithm
Derived session keys
Cryptographic algorithms
Usage
NwkKey
AppKey
AES-ECB
AES-ECB
NwkSKey
JSIntKey
FNwkSIntKey
SNwkSIntKey
NwkSEncKey
AppSKey
AES-CMAC
AES-ECB-1
AES-ECB-1
AES-CMAC
AES-CMAC
AES-CMAC
AES-CTR
AES-CTR
MIC generation for Join Request
Encryption of Join Accept trigged by Join Request
Encryption of Join Accept triggered by Rejoin Request (Type 0, 1, 2)
MIC generation for Rejoin Request (Type 1) and Join Accept
Partial MIC generation for Transmitted Data
Partial MIC for Transmitted Data / Rejoin Request MIC (Type 0, 2)
Encryption of Transmitted Data and Fopt field (Fport =0)
Encryption of Transmitted Data (Fport > 0)

Fig. 9. Key derivation scheme in LoRaWAN v1.1.

However, this improvement makes only a minor contribution to security against our key-recovery attack based on the side-channel analysis. The NwkKey can be recovered in the decryption process of a join-request message. The session keys, JSEncKey, JSIntKey, FNwkSIntKey, SNwkSIntKey, and NwkSEncKey, can be derived from the NwkKey. Some bytes of the AppKey and AppSKey can be recovered.

6 Conclusion

We conducted experiments to extract AES keys that are used to encrypt a frame payload and to calculate the message integrity code (MIC) for the encrypted payload from a real LoRaWAN end-device. Our experiments recovered keys based on a correlation power analysis. The 350 of EM-leakage traces of the payload encryption process can entirely recover the 16-byte payload encryption key while the 140 EM-leakage traces of MIC generation process can recover 12 bytes of the 16-byte MIC generation key. We also achieved 26% of further reduction in the number of traces required to recover keys using a band-pass filtering technique. Furthermore, we showed that our key recovery attack is applicable in real LoRaWAN protocols. Our attack can entirely recover the root key AppKey in LoRaWAN v1.0 and a root key NwkKey in LoRaWAN v1.1. In future work, we will endeavor to recover AES keys from an end-device that supports the real LoRaWAN protocols.

References

1. 3GPP: Standardization of NB-IOT completed, June 2016. http://www.3gpp.org/news-events/3gpp-news/1785-nb_iot_complete
2. Aras, E., Small, N., Ramachandran, G.S., Delbruel, S., Joosen, W., Hughes, D.: Selective jamming of LoRaWAN using commodity hardware (2017). https://doi.org/10.1145/3144457.3144478, http://arxiv.org/abs/1712.02141v1
3. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28632-5_2
4. Butun, I., Pereira, N., Gidlund, M.: Analysis of LoRaWAN V1.1 security: research paper. In: Proceedings of the 4th ACM MobiHoc Workshop on Experiences with the Design and Implementation of Smart Objects, pp. 5:1–5:6. SMARTOBJECTS 2018, ACM, New York, NY, USA (2018). https://doi.org/10.1145/3213299.3213304
5. Clavier, C., Feix, B., Gagnerot, G., Roussellet, M., Verneuil, V.: Improved collision-correlation power analysis on first order protected AES. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 49–62. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23951-9_4
6. Dinu, D., Kizhvatov, I.: EM Analysis in the IoT context: lessons learned from an attack on thread. Cryptology ePrint Archive, Report 2018/076 (2018). https://eprint.iacr.org/2018/076/20180118:125926
7. Fukushima, K., Marion, D., Nakano, Y., Facon, A., Kiyomoto, S., Guilley, S.: Experiment on side-channel key-recovery using a real LPWA End-device. In: 5th International Conference on Information Systems Security and Privacy (ICISSP 2019). pp. 67–74, January 2019. https://doi.org/10.5220/0007259500670074
8. Girard, P.: Low Power Wide Area Networks Security, December 2015. https://docbox.etsi.org/Workshop/2015/201512_M2MWORKSHOP/S04_WirelessTechnoforIoTandSecurityChallenges/GEMALTO_GIRARD.pdf
9. GSMA: Long Term Evolution for Machines: LTE-M (2017). https://www.gsma.com/iot/long-term-evolution-machine-type-communication-lte-mtc-cat-m1/
10. IHS Markit: Number of Connected IoT Devices Will Surge to 125 Billion by 2030, IHS Markit Says (2015). http://www.statista.com/statistics/266210/
11. Itoh, K., Izu, T., Takenaka, M.: Address-bit differential power analysis of cryptographic schemes OK-ECDH and OK-ECDSA. In: Kaliski, B.S., Koç, K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 129–143. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36400-5_11

12. Joye, M., Tymen, C.: Protections against differential analysis for elliptic curve cryptography – an algebraic approach. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2001. Lecture Notes in Computer Science book series (LNCS), vol. 2162, pp. 377–390. Springer, Heidelberg (2001). https://link.springer.com/chapter/10.1007/3-540-44709-1_31
13. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_25
14. Komano, Y., Shimizu, H., Kawamura, S.: Built-in determined sub-key correlation power analysis. Cryptology ePrint Archive, Report 2009/161 (2009). https://eprint.iacr.org/2009/161/20090803:071855
15. Lee, J., Hwang, D., Park, J., Kim, K.H.: Risk analysis and countermeasure for bit-flipping attack in LoRaWAN. In: 2017 International Conference on Information Networking (ICOIN), pp. 549–551, January 2017. https://doi.org/10.1109/ICOIN.2017.7899554
16. LoRa Alliance$^{\text{TM}}$: LoRaWAN$^{\text{TM}}$Specification v1.0.3, November 2018. https://lora-alliance.org/sites/default/files/2018-07/lorawan1.0.3.pdf
17. LoRa Alliance$^{\text{TM}}$: LoRaWAN$^{\text{TM}}$Specification v1.1, November 2018. https://lora-alliance.org/sites/default/files/2018-04/lorawantm_specification_-v1.1.pdf
18. Messerges, T.S., Dabbish, E.A., Sloan, R.H.: Investigations of power analysis attacks on smartcards. In: Proceedings of the USENIX Workshop on Smartcard Technology on USENIX Workshop on Smartcard Technology, pp. 17. WOST 1999, USENIX Association, Berkeley, CA, USA (1999). http://dl.acm.org/citation.cfm?id=1267115.1267132
19. Moukarzel, M., Eisenbarth, T., Sunar, B.: µLeech: A side-channel evaluation platform for IoT. In: 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS), pp. 25–28, August 2017. https://doi.org/10.1109/MWSCAS.2017.8052851
20. Na, S., Hwang, D., Shin, W., Kim, K.H.: Scenario and countermeasure for replay attack using join request messages in lorawan. In: 2017 International Conference on Information Networking (ICOIN), pp. 718–720, January 2017. https://doi.org/10.1109/ICOIN.2017.7899580
21. Sigfox: Sigfox Technology Overview (2017). https://www.sigfox.com/en/sigfox-iot-technology-overview
22. Tawalbeh, L.A., Somani, T.F.: More secure Internet of Things using robust encryption algorithms against side channel attacks. In: 2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA), pp. 1–6, November 2016. https://doi.org/10.1109/AICCSA.2016.7945813
23. Tomasin, S., Zulian, S., Vangelista, L.: Security analysis of LoRaWAN join procedure for Internet of Things networks. In: 2017 IEEE Wireless Communications and Networking Conference Workshops (WCNCW), pp. 1–6, March 2017. https://doi.org/10.1109/WCNCW.2017.7919091
24. Yang, X., Karampatzakis, E., Doerr, C., Kuipers, F.: Security vulnerabilities in LoRaWAN. In: 2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI), pp. 129–140, April 2018. https://doi.org/10.1109/IoTDI.2018.00022
25. Zulian, S.: Security threat analysis and countermeasures for LoRaWAN$^{\text{TM}}$ join procedure. Master's thesis, University of Padova (2016). http://tesi.cab.unipd.it/53210/