Evaluating the cost of beyond AES-128 LoRaWAN security

Phithak Thaenkaew
University of Mons
Mons, Belgium
phithak.thaenkaew@umons.ac.be

Bruno Quoitin
University of Mons
Mons, Belgium
bruno.quoitin@umons.ac.be

Ahmed Meddahi
IMT Nord Europe
Lille, France
ahmed.meddahi@imt-nord-europe.fr

***Abstract*—LoRaWAN is one of the most popular Internet of Things radio communication technologies since it allows data to be transmitted over long distances with low power consumption on unlicensed ISM bands. Devices used with LoRaWAN are often constrained for reasons of manufacturing cost and to save energy as they are most often battery-powered. As a consequence, security mechanisms chosen for such devices must be lightweight. The LoRaWAN standard relies on AES-128 for encryption and authentication, using a pre-shared key.**

With the increasing amount of computational power at hand and the advent of quantum cryptography, the use of short AES keys without renewal for long periods of time is a potential weakness. However, using longer keys in LoRaWAN impacts end devices in terms of processing time and energy consumption. It might also require adaptations in the protocol design. This paper investigates the cost of using different AES key sizes with different payload sizes on an off-the-shelf LoRaWAN platform. Our results show that costs in terms of delay and energy consumption are moderate and using longer key sizes is a practical solution to increase the security of LoRaWAN.

***Index Terms*—LoRaWAN, LoRa, IoT, IIoT, Security**

I. Introduction

It is well known that nowadays Internet of Things (IoT) technology plays an increasingly important role in our daily life. Many different devices are connected to the Internet to share information with other things. Various IoT communication mechanisms have been developed to suit different types of applications. For example, multiple Low-Power Wide Area Network (LPWAN) technologies exist such as LoRaWAN, NB-IoT and Sigfox. LoRaWAN is one of the most popular for three main reasons: it uses low power, it can transmit data over a distance of several kilometers and it uses the license-free ISM band. However, since it is expected that most LoRaWAN devices are battery-powered, increasing the battery lifetime requires keeping the node power consumption at a minimum. The use of constrained resources affects various aspects of communication, including security. In other words, very strong and sophisticated security mechanisms cannot be applied. For the LoRaWAN standard, this results in security mechanism based on AES-128 encryption and authentication relying on a pre-shared key (PSK).

The security of the LoRaWAN technology has been the subject of several studies. Multiple vulnerabilities and attacks have been described [1, 2], such as denial-of-service, disclosure and modification of message content, as well as battery exhaustion. Automated security protocol verification tools such as Scyther [3] have been used to uncover such vulnerabilities. For example, known vulnerabilities of version 1.0 of LoRaWAN Over-the-Air Authentication (OTAA) were detected [4], while version 1.1 no longer showed these vulnerabilities.

In a recent survey of LPWAN security [5], the length of symmetric cryptography keys is questioned, especially since such keys are used for extended periods of time, typically during the whole lifetime of devices. Using AES with 128-bits keys might not be strong enough due to increased computing power available in the future. Furthermore, the advent of quantum computing questions the strength of today's cryptographic algorithms. In particular, Grover's algorithm [6] can be applied to break AES-128 with resources estimated to 2,953 qubits [7]. Figure 1 based on IBM's roadmap for scaling quantum technology[1] shows how the number of qubits is expected to grow from 2019 onward. Looking at the trend, the number of qubits required to make such attack practical is expected to be reached in the near future.

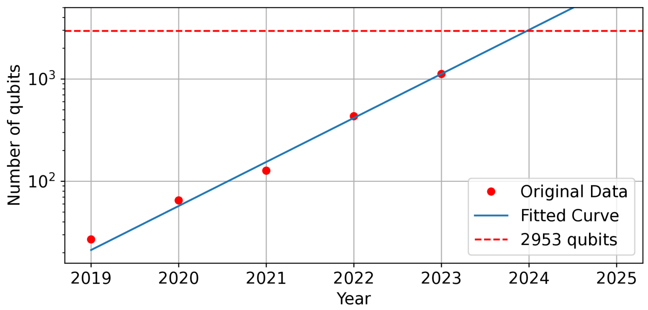

Fig. 1: IBM's roadmap for scaling quantum technology.

The use of AES on embedded systems and its impact in terms of resource usage has been studied in a few papers, some of them focusing on mobile devices [8], others on IoT devices [9, 10]. None of them has considered that question in the context of LoRaWAN. Even though the power consumption of LoRaWAN devices has been studied [11], the impact of varying the AES key size has not been considered yet. Therefore, in this paper, we explore what changes are required in LoRaWAN to make

[1] https://research.ibm.com/blog/ibm-quantum-roadmap

978-1-6654-8544-9/22/$31.00 ©2022 IEEE

use of longer AES key sizes and evaluate experimentally the impact on end devices performance in terms of processing time and energy consumption.

The paper is organized as follows. We discuss related work on Section II. Section III provides the necessary background about LoRaWAN and its authentication mechanism. Section IV describes our evaluation methodology and exposes our measurement results. Finally, we conclude in Section V.

II. Related Work

This section explores existing work related to LoRaWAN security, with a focus on the link with energy consumption.

Since no clear guidelines exist in the LoRaWAN standard to secure the network core, new secure LoRaWAN architectures have been proposed [12, 13]. Noting that the gateway is the most exposed part of the infrastructure as it lies outside the network core, Oniga and his colleagues [12] connect the gateways to the core through VPNs. Moreover, every communication within the network core, e.g. between network and application servers, is secured with TLS and mutual authentication based on a Public Key Infrastructure (PKI) and Certificate Authority (CA). Another secured LoRaWAN backend [13] is proposed focusing on the communication layer between stack servers. Their proposal named Server Session Key Generation (S2KG) relies on using Ellipitic Curve Cryptography (ECC) to derive network sessions keys.

LoRaWAN security evaluation testbed have also been proposed. For example, ChirpOTLE [14] is used for verifying a novel Adaptive Data Rate (ADR) spoofing attack and the vulnerability of missing beacon authentication in Class B operation. It allowed the authors to propose countermeasures for both attacks by revising the LoRaWAN specification. Another proposal [15] combines standalone LoRaWAN transceivers with software-defined radio (SDR) and GNU Radio. It allows them to reproduce a man-in-the-middle attack.

The evaluation of symmetric-key cryptography on resource-constrained devices has been the topic of multiple evaluations. For example, a comparison of DES, 3DES and AES on mobile devices (PDA) was performed by Rif'a-Pous and Herrera-Joancomart [8]. In [16], the processing time of multiple lightweight hardware substitution-permutation network (SPN) block ciphers were compared, with the objective of using them on low-resource devices. The analysis of AES latency and energy consumption on Contiki-based IoT devices was proposed in [9], using the Texas Instruments ARM-based CC2650 on which three AES implementations were tested: Contiki's own built-in AES, tinyAES and B-Con's AES. The former outperforming the others in terms of duration and energy consumption. Using a similar platform, the Texas Instruments CC1310, [10] compares software and hardware AES implementations, concluding that hardware implementations indeed reduce the duration and energy consumption. However, even with hardware AES, software remains involved for the application over multiple blocks. Tsai et al propose a hardware low-power AES data encryption architecture (LPADA) [17] for LoRaWAN. The crust of the proposal is to implement AES substitution through a low-power lookup-table and better manage power distribution of unused AES logic along its different rounds.

An analytical model of LoRaWAN energy performance was proposed [11], predicting a 1-year battery lifetime should be achieved by an end-device running on a 2400 mAh battery with 5 min message sending interval while it would tend asymptotically to about 6 years with increasingly larger intervals. However, that study does not focus on the cost of security primitives.

To the best of our knowledge, no literature discusses the impact of using longer AES key sizes with LoRaWAN end-device.

III. Background

LoRaWAN is a Low Power Wide Area Network (LPWAN) technology. It allows the transmission of small packets over long-distance by resource-constrained devices. It combines wireless access links using the LoRa physical layer with an IP-based network core. The topology of a LoRaWAN network is illustrated in Figure 2. There are 4 main components in this architecture. First, *end devices* (ED), shown on the left, typically consist of a radio transceiver and antenna combined with a micro-controller that takes care of sampling, processing and transmitting sensor data. End devices are often battery powered. Second, *gateways* (GW) are used by end devices to access the LoRaWAN network. Gateways also consist of a radio transceiver and a microprocessor. However, as opposed to the end nodes, gateways are typically mains powered and connected to the Internet through a wired or cellular network. It is typical (and desirable) that the transmission of an end device be captured by multiple gateways. The last two components, the *Network* (NS) and *Application servers* (AS), are part of the LoRaWAN core and communicate with each other through IP-based communications.

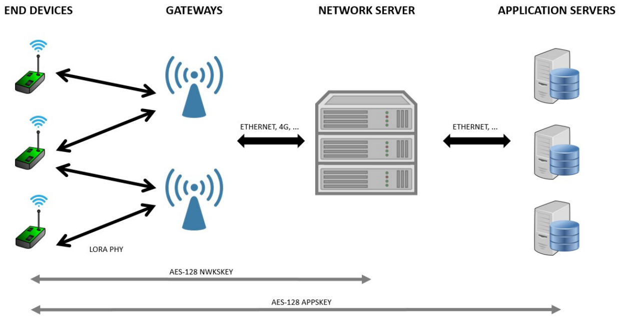

Fig. 2: LoRaWAN network architecture.

The whole LoRaWAN network allows end devices to send data to client applications. However, there is no direct communication between them. The end device data is first received by gateways, then processed by network servers and finally made available on the application servers. Client applications can then retrieve the received data through e.g. web APIs or the MQTT protocol. The communication is bidirectional. This means clients can also schedule the transmission of data down to end devices.

Since end devices spend most of their time in sleep mode, they only wake up when they need to transmit data (*uplink*) or to receive data (*downlink*). Downlink transmissions can only occur after an uplink transmission to allow the network to learn when the device is awaken. For this purpose, just after an uplink transmission, an end device is allowed to schedule up to two short periods of times, named *receive windows*, where it will remain in listening mode.

A. LoRa physical layer

LoRa is a proprietary radio modulation technique due to Semtech. It is used as the physical layer of a LoRaWAN network for communications between the end devices and gateways. LoRa transmissions are performed over Industrial, Scientific and Medical (ISM) bands which can be used unlicensed worldwide, the exact frequency bands varying by region.

LoRa modulation is *Chirp Spread Spectrum* (CSS), a technique where symbols correspond to signals whose frequency varies linearly with time in a cyclic manner. Data is encoded by modifying the starting frequency of such cycle. One parameter of LoRa is the *Spreading Factor* (SF). It influences the number of bits that can be encoded in a symbol and at the same time the duration of symbols. Larger spreading factor values make the signal easier to recover, hence make transmissions possible over longer distances. Smaller values reduce the transmission time of LoRa frames. The transmission range of LoRa varies from up to a few kilometers in urban areas to over several kilometers in rural areas or direct line of sight [18].

Transmissions using different spreading factors are orthogonal. This means that such transmissions, occurring simultaneously will not collide. LoRaWAN gateways are typically capable of demodulating multiple orthogonal transmissions received at the same time.

B. LoRaWAN security

To provide secure communications, LoRaWAN relies on the Advanced Encryption Standard (AES), a symmetric-key block cipher using the principle of substitution-permutation network. Although AES supports key sizes of 128, 192 and 256 bits, the shorter size was adopted in the standard. The reason is the key size affects the number of transformation rounds to be computed: 10, 12 and 14 rounds for keys of 128, 192 and 256 bits, respectively. In LoRaWAN, AES is used for encryption using the AES-CCM* scheme but also for computing Message Integrity Code (MIC), using AES-CMAC.

LoRaWAN security is organized in two layers, as can be seen at the bottom of Figure 3. First, confidentiality is achieved from the end-device to the application server by encrypting the message payload thanks to AES-CCM*, using an *application session key* (`AppSKey`). Second, data origin authentication, integrity and replay protection is ensured between the end device and the network server, thanks to the inclusion of a frame counter (`FCnt`) and using AES-CMAC to compute a MIC with a *network session key* (`NwkSKey`).

Before end devices are able to transmit and receive data securely through a LoRaWAN network, they need to be activated. This process allows end devices to derive the necessary session keys, to obtain a short address (`DevAddr`) and some network parameters. There are currently two methods to activate an end device: *Activation by Personalization* (ABP) and *Over-the-Air Activation* (OTAA). Essentially, with ABP, devices are provisioned with session keys at configuration time, while with OTAA, the session keys are dynamically derived from a *per-device root key* (`AppKey`). OTAA is the preferred activation method since it makes re-keying possible. Moreover, it allows to work with different networks as the `NwkSKey` will be re-generated upon joining.

C. Over-the-Air Activation

Before its activation with OTAA, an end device must be configured with its 64-bits identifier (`DevEUI`), the application 64-bits identifier (`AppEUI`) and its per-device 128-bits root key (`AppKey`). These parameters are typically configured by the device manufacturer. The network server stores a pair (`DevEUI`, `AppKey`) for every device provisioned in the network.

The OTAA procedure is illustrated in Figure 3. We now proceed to detail every step.

1) The end device sends a `Join-Request` message to the network server. This message contains the `AppEUI` and `DevEUI` as well as a `DevNonce`. The latter is a randomly generated number to prevent replay attacks. The `Join-Request` message is sent in cleartext but appended with a MIC calculated by applying AES-CMAC on the message, with the `AppKey`.
2) When the network server receives the `Join-Request` message, it checks the `DevNonce` has not been used previously. To this end, it maintains the list of last `DevNonce` used by each end device. The network server authenticates the end device by re-calculating and comparing the MIC. If it corresponds to the received MIC, the end device is authenticated. This indeed proves the end device knows the `AppKey`. The network server then generates a 32-bits address (`DevAddr`) for the end device and a randomly generated `AppNonce`.
3) The network server constructs a `Join-Accept` message containing the `DevAddr`, the `AppNonce`, a network idenfitier (`NetID`) and some network settings. Over this message, a MIC is generated by AES-CMAC, using the `AppKey`. The `Join-Accept` message itself is encrypted by AES-CCM* with the `AppKey`. The network server sends the message and MIC back to the end device.
4) Both the end device and the network server now share the same `AppNonce` and `DevNonce`. Together with the `AppKey`, they are used to derive the application session key (`AppSKey`) and network session key (`NwkSKey`). The network server sends the `AppSKey` and `DevAddr` to the application server.

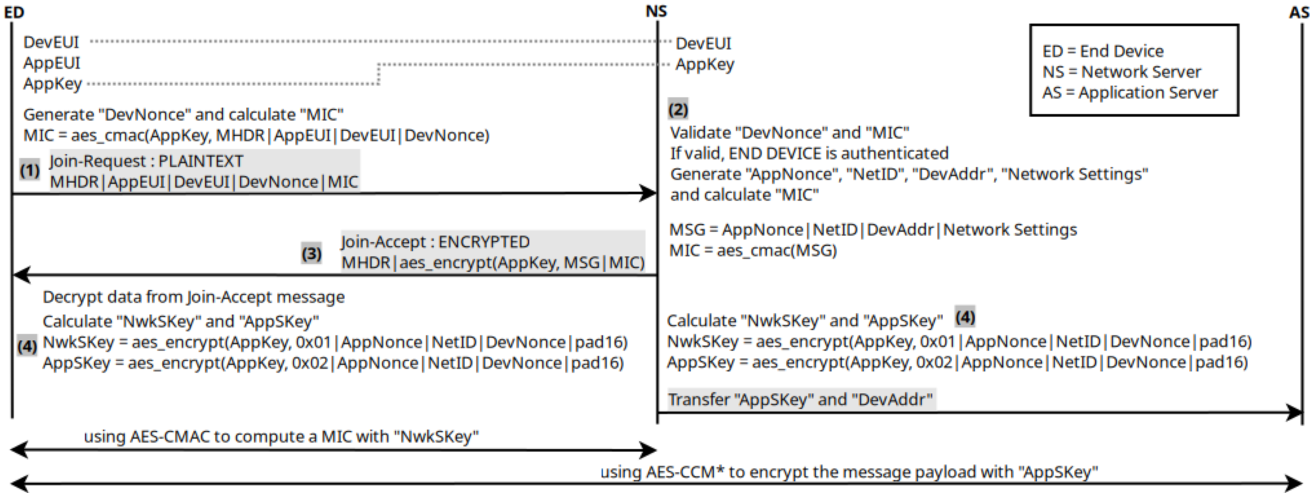
ED
NS
AS
DevEUI
AppEUI
AppKey
DevEUI
AppKey
ED = End Device
NS = Network Server
AS = Application Server
Generate "DevNonce" and calculate "MIC"
MIC = aes_cmac(AppKey, MHDR|AppEUI|DevEUI|DevNonce)
(1) Join-Request : PLAINTEXT
MHDR|AppEUI|DevEUI|DevNonce|MIC
(2)
Validate "DevNonce" and "MIC"
If valid, END DEVICE is authenticated
Generate "AppNonce", "NetID", "DevAddr", "Network Settings"
and calculate "MIC"
MSG = AppNonce|NetID|DevAddr|Network Settings
MIC = aes_cmac(MSG)
(3) Join-Accept : ENCRYPTED
MHDR|aes_encrypt(AppKey, MSG|MIC)
Decrypt data from Join-Accept message
Calculate "NwkSKey" and "AppSKey"
(4) NwkSKey = aes_encrypt(AppKey, 0x01|AppNonce|NetID|DevNonce|pad16)
AppSKey = aes_encrypt(AppKey, 0x02|AppNonce|NetID|DevNonce|pad16)
Calculate "NwkSKey" and "AppSKey" (4)
NwkSKey = aes_encrypt(AppKey, 0x01|AppNonce|NetID|DevNonce|pad16)
AppSKey = aes_encrypt(AppKey, 0x02|AppNonce|NetID|DevNonce|pad16)
Transfer "AppSKey" and "DevAddr"
using AES-CMAC to compute a MIC with "NwkSKey"
using AES-CCM* to encrypt the message payload with "AppSKey"

Fig. 3: LoRaWAN v1.0 Over-the-Air Activation

OTAA v1.0 has a security weakness [2] in the `AppSKey` calculation from the root key that allows the network servers to access the encrypted data between end devices and application servers. In OTAA v1.1, a new entity, named *join server* (JS) is responsible for the session key establishment. It then share the `NwkSKey` with network servers and the `AppSKey` with application servers. We do not discuss this version further as, from an end device perspective, it does not change the required resources.

IV. Experimental evaluation

We designed a methodology based on a specific testbed and firmware to quantify the impact of changing the AES key size when applying LoRaWAN encryption, decryption and MIC calculation on payloads of different sizes. We rely on the following performance metrics: duration and energy consumption of AES encryption and MIC operations. Our measurements are performed on a typical IoT device supporting a LoRaWAN stack.

A. Experimental Platform

To perform our measurements, we setup a small testbed, as illustrated on Figure 4. The Device Under Test (DUT) is a LoRaWAN development board based on an ultra-low-power ARM Cortex-M0+ MCU (STM32L072CZ) running at 32 MHz. It is combined with a Semtech SX1276 LoRa transceiver. The ARM MCU has no hardware cryptography support, which means that the crypto primitives we test are completely implemented in software in the LoRaWAN stack.

To measure the energy consumption of the DUT, we rely on a JouleScope JS110 precision DC energy analyzer from JetPerch. This device allows to simultaneously measure the power supply voltage and current of the DUT, and therefore estimate its power consumption. It does this at very high rate (2 MSamples/s), with a bandwidth of 250 kHz. Moreover it has a very high dynamic current range spanning nanoAmps to Amps (9 orders of magnitude), a required feature to measure the energy consumption of devices with very low power modes.

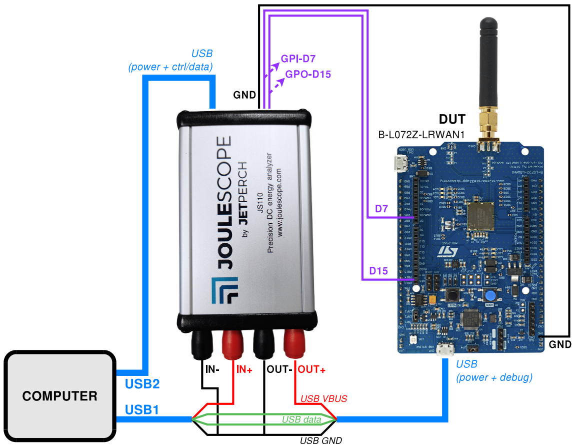

Fig. 4: Wiring diagram of the measurement testbed.

As shown on Figure 4, the DUT power is provided through its USB port. We interrupted it to make it go through the JouleScope input and output ports. In addition to this, we connected GPIO ports of the DUT to the JouleScope digital inputs. These GPIO ports are under control of the test firmware and allow to mark with high precision specific times in the voltage/current trace collected by the JouleScope. These marks allow to identify regions of interest and extract the related samples.

B. Methodology

To achieve our objective of measuring the power consumption and processing time induced by the cryptography primitives used in the LoRaWAN stack, we developed a specific test firmware for the STM32 MCU. We relied on the Mbed OS[2], using the LoRaMacCrypto library. The toolchain was Mbed CLI2. We selected the following three functions in the LoRaMacCrypto for analysis: `encrypt_payload` which performs the AES-CCM* encryption of LoRaWAN payloads,

[2] https://github.com/ARMmbed/mbed-os

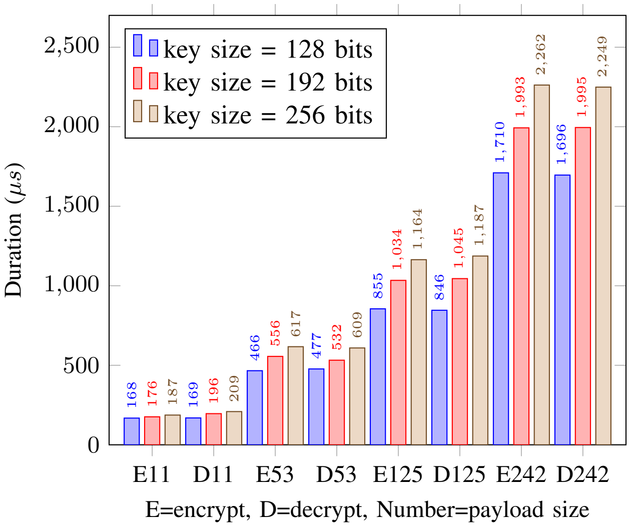

Fig. 5: Duration of AES encryption and decryption

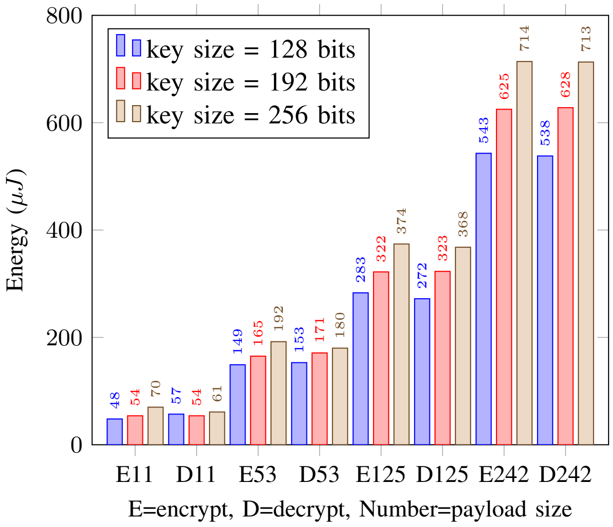

Fig. 6: Energy consumption of AES encryption and decryption

decrypt_payload which does the corresponding decryption, and compute_mic which calculates the MIC using AES-CMAC. These functions were instrumented to toggle the GPIO pins described in Section IV-A and produce the markers in the captured trace upon entry or exit of the functions. In addition to this we use an Mbed OS timer to measure the time spent in the functions. We validated that the measured time matches the interval between the markers.

We perform our measurements for the 3 possible AES key sizes : 128 bits, 192 bits and 256 bits. Moreover, we use 4 different payload sizes: 11 bytes, 53 bytes, 125 bytes and 242 bytes. The sizes of payload are chosen from the maximum uplink user payload size [18].

C. Experimental Result

The results are divided into two main sections. First, the measured processing times are reported in Figure 5 for the AES encryption and decryption and Table I (1st row) for the MIC calculation. Then, the energy consumption (in Joules) is reported in Figure 6 and Table I (2nd row). Measurements were performed 10 times for every combination of key size and payload size. We report the average of these values.

We first observe that the processing time and energy consumption increase linearly with the payload size, regardless of the AES key size. For example, the processing time for AES128 encryption goes from $168\mu s$ (E11) with the smallest payload to $1,710\mu s$ with the largest one (E242). This corresponds respectively to energy consumption equal to $48\mu J$ and $543\mu J$. Second, since using larger key sizes increases the number of rounds of the AES block cipher, we expected that the processing time and energy consumption would increase in the same proportion. This is indeed what we observe in the results for Encrypt and Decrypt. For example, looking at the largest payload, the processing time (E242) jumps from $1,710\ \mu s$ to $2,262\mu s$, a 32% increase, when changing from AES128 (10 rounds) to AES256 (14 rounds). The results for Encrypt and Decrypt are almost the same. This is also expected as AES encryption and decryption are the reverse of each other and the encrypt_payload and decrypt_payload functions share a common code base. For the MIC results, although AES-CMAC can be used with AES192 and AES256 [19], the LoRaMacCrypto implementation only works with 128-bits keys. For this reason, we could only measure that function with the shortest key size.

Although the increase in processing time and energy consumption seem significant, they remain very limited when compared to the time and energy required for a complete data transmission. For example, using the SF7 spreading factor with 500 kHz bandwidth, the transmission time (time-on-air) of a 242 bytes payload amounts to ≈ 90.5 ms. The time dedicated to payload encryption represents only 2.5% of the transmission time. With higher spreading factors or narrower bandwidths, the impact is even smaller.

We observed limited variance across our processing time and energy measurements for a given pair of input parameters. We ascribe such variations to two factors. First, our test firmware does not disable interrupts during the execution of the functions which sometimes implies small increases in measured processing time, and as a result increased estimated energy consumption. Second, the USB power supply voltage provided by the computer exhibited some voltage ripple. As the energy is calculated as the integral of the product of the current and voltage samples, this voltage variation causes a bit of noise in the resulting energy estimate. Finally, we measured the power consumption at the USB port of the DUT, which implies that other components of the development board were also drawing current, such as the debug USB interface (ST-LINK). These variations were however very limited and do not affect the conclusion of our study.

V. Conclusion

The current version of LoRaWAN relies on AES with 128-bits key size to secure its communications. AES is used for

TABLE I: Duration and energy consumption of MIC calculation

	Payload Size			
	11 Bytes	**53 Bytes**	**125 Bytes**	**242 Bytes**
Duration (μs)	621	919	1,353	2,218
Energy (μJ)	186	287	430	692

Note: *AES key size 128 bits*

the confidentiality of data, using the AES-CCM* mode, and to provide origin authentication, message integrity and prevent replay, by using AES-CMAC. Our objective in this paper was to evaluate the cost of using longer AES key sizes with the perspective to strengthen LoRaWAN's security. The cost of the cryptography operations is indeed important to consider on resource-constrained devices. To this end, we setup an evaluation testbed composed of a LoRaWAN embedded platform running a custom firmware. We measure the processing time and energy consumption of targeted LoRaWAN security primitives under different combinations of AES key sizes and payload sizes.

From the experimental results we observe that the considered metrics indeed increase with key and payload sizes. However, the impact is moderate, making using larger AES key size a practical solution. For example with the largest payload size of 242 bytes, the AES encryption duration increased by about 32 % when changing from AES128 to AES256. The resulting additional energy remains however very low compared to the cost of other operations such as radio communications.

Using a larger AES key size does not imply changes in the LoRaWAN frame formats. However, the network stack on the end device and the network and application servers need to be updated. In our further work, we plan to completely implement such changes. In addition, we envision an additional activation method based on using asymmetric cryptography and certificates to allow for stronger authentication by relying on Elliptic-Curve Cryptography (ECC). We expect the incurred energy cost to remain acceptable for resource-constrained devices.

References

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

[2] S. Tomasin, S. Zulian, and L. Vangelista, "Security Analysis of LoRaWAN Join Procedure for Internet of Things Networks," in *2017 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, 2017.

[3] C. Cremers, "Scyther - Semantics and Verification of Security Protocols," Ph.D. dissertation, Eindhoven University of Technology, 2006, ISBN: 978-90-386-0804-4.

[4] M. Eldefrawy, I. Butun, N. Pereira, and M. Gidlund, "Formal security analysis of LoRaWAN," *Computer Networks*, vol. 148, 2019, ISSN: 1389-1286.

[5] F. L. Coman, K. M. Malarski, M. N. Petersen, and S. Ruepp, "Security Issues in Internet of Things: Vulnerability Analysis of LoRaWAN, Sigfox and NB-IoT," in *2019 Global IoT Summit (GIoTS)*, 2019.

[6] L. K. Grover, "A Fast Quantum Mechanical Algorithm for Database Search," in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, ser. STOC '96, Philadelphia, Pennsylvania, USA: ACM, 1996, ISBN: 0897917855.

[7] M. Grassl, B. Langenberg, M. Roetteler, and R. Steinwandt, *Applying Grover's algorithm to AES: quantum resource estimates*, 2015. arXiv: 1512.04965.

[8] H. Rifà-Pous and J. Herrera-Joancomartı, "Computational and energy costs of cryptographic algorithms on handheld devices," *Future Internet*, vol. 3, no. 1, 2011, ISSN: 1999-5903.

[9] B. Tsao, Y. Liu, and B. Dezfouli, "Analysis of the Duration and Energy Consumption of AES Algorithms on a Contiki-Based IoT Device," in *Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems (MobiQuitous)*, Houston, Texas, USA: ACM, 2019, ISBN: 9781450372831.

[10] C.-W. Hung and W.-T. Hsu, "Power Consumption and Calculation Requirement Analysis of AES for WSN IoT," *Sensors*, vol. 18, no. 6, 2018, ISSN: 1424-8220.

[11] L. Casals, B. Mir, R. Vidal, and C. Gomez, "Modeling the Energy Performance of LoRaWAN," *Sensors*, vol. 17, no. 10, 2017, ISSN: 1424-8220.

[12] B. Oniga, V. Dadarlat, E. De Poorter, and A. Munteanu, "Analysis, design and implementation of secure LoRaWAN sensor networks," in *2017 13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*, 2017.

[13] K.-L. Tsai, F.-Y. Leu, L.-L. Hung, and C.-Y. Ko, "Secure Session Key Generation Method for LoRaWAN Servers," *IEEE Access*, vol. 8, 2020.

[14] F. Hessel, L. Almon, and F. Álvarez, "ChirpOTLE: A Framework for Practical LoRaWAN Security Evaluation," in *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, ser. WiSec '20, Linz, Austria: ACM, 2020, ISBN: 9781450380065.

[15] O. Pospisil, R. Fujdiak, K. Mikhaylov, H. Ruotsalainen, and J. Misurec, "Testbed for LoRaWAN Security: Design and Validation through Man-in-the-Middle Attacks Study," *Applied Sciences*, vol. 11, no. 16, 2021, ISSN: 2076-3417.

[16] M. Knežević, V. Nikov, and P. Rombouts, "Low-latency encryption – is "lightweight = light + wait"?" In *Cryptographic Hardware and Embedded Systems – CHES 2012*, E. Prouff and P. Schaumont, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, ISBN: 978-3-642-33027-8.

[17] K.-L. Tsai, F.-Y. Leu, I. You, S.-W. Chang, S.-J. Hu, and H. Park, "Low-Power AES Data Encryption Architecture for a LoRaWAN," *IEEE Access*, vol. 7, 2019.

[18] Semtech Corporation, *LoRa and LoRaWAN : A Technical Overview*, Dec. 2019.

[19] M. Dworkin, *Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication*, NIST SP-800-38B, 2005.