Received: 12 October 2018 | Revised: 15 April 2019 | Accepted: 26 April 2019
DOI: 10.1002/cpe.5352

WILEY

SPECIAL ISSUE PAPER

Decentralized LPWAN infrastructure using blockchain and digital signatures

Arnaud Durand | **Pascal Gremaud** | **Jacques Pasquier**

Department of Informatics, University of Fribourg, Fribourg, Switzerland

Correspondence
Arnaud Durand, Department of Informatics, University of Fribourg, 1700 Fribourg, Switzerland.
Email: arnaud.durand@unifr.ch

Summary

This paper introduces a fully decentralized low-power wide-area network (LPWAN) infrastructure for the Internet of Things (IoT) using the LoRa protocol. While global LPWANs typically require roaming agreements between network providers and a trusted third party for server resolution, we propose a trustless model where the network servers are resolved using a blockchain application. Since LoRaWAN relies on symmetric cryptography, we also propose a new security model that adds non-repudiation using digital signatures. This paves the way for linking devices to decentralized applications. We finally analyze the impact of this new model on message size and energy requirements.

KEYWORDS

blockchain, distributed systems, Internet of Things, LoRaWAN

1 | INTRODUCTION

Low power communication is a major milestone for the Internet of Things (IoT). Low-Power Wide-Area Network (LPWAN) technologies seek to provide a large coverage area and a long battery life at the cost of reduced bandwidth compared to legacy cellular networks. LTE network carriers are actively upgrading to newer standards such as NB-IoT or LTE-M to fill this gap. However, this type of service is restricted to telecom operators buying expensive licences in order to use cellular frequencies. A promising alternative is LPWAN running on unlicensed frequency bands. This is characterized by long-range sub-gigahertz radio links and star topologies.[1] In addition to services offered by network providers, unlicensed bands enable individuals and organizations to run their own network without a third party. This creates a rich ecosystem where organizations can build conglomerates of networks in order to reach a wider, potentially global coverage. Similarly, it is possible to crowd-source parts of the network such as radio gateways—transceivers that sit between end-devices and a wide area network. An example of such a network is The Things Network (TTN),[2] a global LoRaWAN data network that leverages radio gateways provided by its community. In TTN, gateways belong to users while network backend servers are managed by TTN itself. We push the crowd-sourced idea further by trying to build a global and fully decentralized IoT network. Among the advantages of a decentralized network are a high resilience against many threats and improved scalability.

We propose a solution based on the LoRaWAN protocol and build a prototype of a decentralized IoT network using a blockchain-connected packet forwarding application and off-the-shelf hardware. There have been previous attempts to conceptualize such a decentralized network,[3] but to the best of our knowledge, we released the first practical implementation of such a system using LoRaWAN.[4] Bezahaf et al then proposed an alternative LoRa scheme that solves the fair exchange problem,[5] which is complementary to our previous work. Blockchain technology is a key element of the system, since it makes it possible to build a shared globally synchronized key-value store of network “join” servers without the involvement of a trusted party. These servers are capable of activating a particular end-device, a remote node communicating through a LoRa radio link. This article is an extension of our previous research on resilient crowd-sourced LPWAN infrastructure using blockchain.[4] In this article, we take the idea a step further by highlighting the benefits digital signatures would bring to this model and we analyze its feasibility in terms of requirements and transmission overhead.

The remainder of this paper is structured as follows. In Section 2, we introduce both the LoRaWAN protocol and the blockchain technology. We then describe the process needed to achieve network decentralization in Sections 3.1 to 3.3. From this high-level description, we present our prototype implementation, which is freely available to download. We then propose an alternative model using asymmetric cryptography

TABLE 1 The LoRaWAN protocol stack

Application Layer	
LoRaWAN	MAC
LoRa	PHY
EU868 | US915 | …	RF

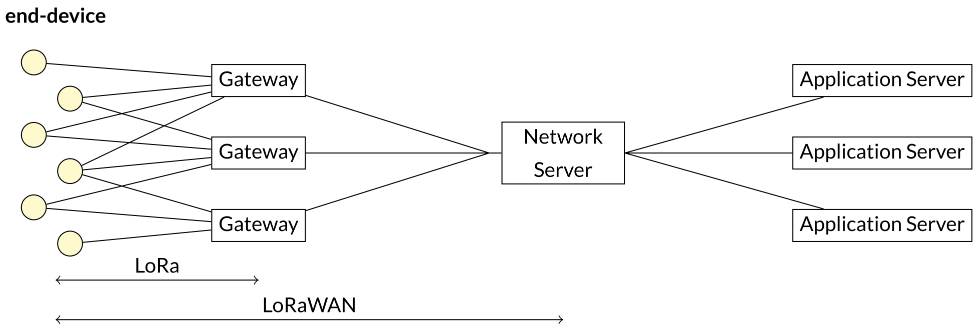

FIGURE 1 A typical LoRaWAN infrastructure. The network server is at the center of the star topology

in Section 4. From this new perspective, we quantify some of the outcomes to evaluate its applicability. We conclude in Section 5 with some insights about the potential decentralized applications when shifting to a new security model.

2 | BACKGROUND

2.1 | LoRa/LoRaWAN

LoRaWAN is a LPWAN protocol built on top of LoRa, a proprietary physical layer (PHY) radio developed by Semtech using chirp modulation. As shown in Table 1, LoRaWAN provides the media access control (MAC) layer in order for LoRa devices to connect to a wide area network, such as the Internet. There exist other LoRa-based protocols proposed by the industry, such as Symphony Link and D7A-Lora. Symphony Link is a proprietary solution from Link Labs that claims to improve scalability over LoRaWAN, while D7A-LoRa is an adaptation of the DASH7 protocol on top of LoRa. The LoRaWAN MAC layer is defined in the LoRaWAN 1.1 Specification.[6]

A typical LoRaWAN network consists of end-devices, gateways, a network server, and application servers. This is illustrated in Figure 1. End-devices (sensors or actuators) communicate with gateways using radio transceivers. Gateways (aka concentrators) forward unprocessed LoRaWAN messages between the air and the network server without inspection. The network server terminates the LoRaWAN MAC layer for the end-devices. For better scalability, the network server could consist of several machines serving different roles. Note that communication with application servers is outside of the scope of the LoRaWAN specification. In the case of an infrastructure belonging to a commercial provider, both the gateways and the network server are provided by the network operator. Application servers run on the behalf of the device owner to preserve data confidentiality. For private networks, a single entity manages the whole infrastructure.

In the case of TTN, gateways are provided by volunteers. In this scenario, only gateways are crowd-sourced; the network backbone implementing services, such as message routing, is provided by TTN's own infrastructure.

2.1.1 | Security

LoRaWAN security is provided by both symmetric message integrity check and encryption. End-devices use the *AppKey* root key (LoRaWAN 1.0) or *AppKey* and *NwkKey* root keys (LoRaWAN 1.1), which are provisioned before deployment. The application key (*AppKey*) is used for data confidentiality, while the network key (*NwkKey*) is used for message integrity and the confidentiality of MAC commands. Having two different keys enables the authentication and delegation with the network provider without compromising the confidentiality of the application layer. Due to the sole use of symmetric cryptography, it is not possible to use digital signatures on the LoRaWAN MAC layer.

2.1.2 | Roaming

There are two possible types of roaming in LoRaWAN: passive roaming and handover roaming. Passive roaming consists of the redirection of packets from a Forwarding Network Server (fNS) to the Home Network Server (hNS), which is illustrated in Figure 2. Note that we distinguish Join Server (JS) and Application Server (AS) from hNS for better consistency with the LoRaWAN Backend Interfaces.[7] We will, however, simplify and describe a monolithic network server incorporating the Join Server from now on. When communicating in a passive roaming scenario, it is totally invisible to end devices that they are connected to an fNS. On the other hand, a serving network server actively communicates with

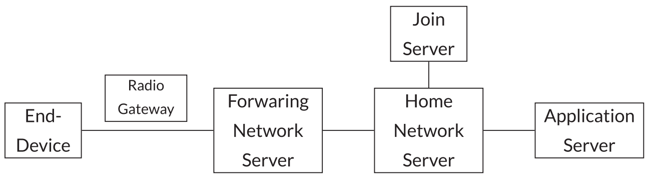

FIGURE 2 A LoRaWAN network model performing passive roaming

end-devices in handover roaming scenario by sending MAC commands. Passive roaming is supported in both LoRaWAN 1.0.x and 1.1, while handover roaming was introduced in LoRaWAN 1.1. Roaming procedures are extensively described in LoRaWAN Backend Interfaces 1.0.[7]

2.2 | Blockchain

Distributed ledgers represent a breakthrough in decentralized permissionless (or trustless) systems, namely blockchain databases. A blockchain enables trusting the output of a system without trusting anyone in particular. The blockchain technology was first described in the Bitcoin white paper.[8] Bitcoin can be described as a distributed, public, and globally synchronized ledger containing cash, (ie digital tokens), transactions. Due to the cryptographic primitives it is based on, there are strong security assumptions against transaction tampering such as alteration, reversing, or reordering. Bitcoin popularized secure decentralized money transactions, ie, cryptocurrencies, and researchers quickly realized that this technology opened the way to much wider applications by the means of smart contracts. Smart contracts are self-executing software where rules and agreements are expressed as computer code. They were theorized before Bitcoin,[9] but had no underlying architecture before blockchains. They are the digital equivalent of a contract. Compared to other distributed applications, smart contracts enable truly permissionless consensus-based decentralized computing.

The use of blockchain in the IoT is an ongoing research topic. Applicability includes, but is not limited to, traceability (ie, in supply chain or to guarantee provenance, authenticity, and the compliance of end products), asset sharing, and autonomous marketplaces.[10-12] We expect a lot of novel uses to emerge in the near future.

3 | DECENTRALIZING LORAWAN

This section describes the steps necessary to decentralize a LoRaWAN network. Until Section 4, the proposed method maintains compatibility with existing LoRaWAN devices, an important concern when upgrading an existing infrastructure or when using radios with integrated LoRaWAN stack.

3.1 | Join server registration and resolution

A global LoRaWAN network consists of either (1) backbone servers from a single provider with a global coverage (eg, TTN) or (2) a group of network servers with roaming agreements (ie, business agreements for commercial operators). In the latter, the hNS address is resolved using DNS (see LoRaWAN Backend Interfaces[7]) and supports alternative addresses to allow for high-availability and geo-redudancy. Moreover, the referred DNS is managed exclusively by the LoRa Alliance, a non-profit organization whose goal is to standardize the LoRaWAN network. In our approach, the smart contract provides implicit roaming agreements and hNSs are resolved using a blockchain application. As opposed to relying on existing network server infrastructures from a provider, it implies that participants run their own network server in addition to the application server.

In order for fNSs (passive roaming) and serving network servers (handover roaming) to unambiguously identify which network server an end-device belongs to, the end-device provides a 64-bit join identifier (*JoinEUI*; previously *AppEUI* in LoRaWAN < 1.1) beforehand during the join procedure. This handshake is described in more details in Section 3.2. In the proposed solution, *JoinEUI*s are registered in a smart contract. Analogically to DNS servers, the smart contract acts as a domain name service for gateways, translating *JoinEUI*s to LoRaWAN nework server addresses. This is achieved with the help of an *addressable* contract which contains mappings between registrants and network addresses. On top of that, *JoinEUI*s are registered and stored in a registry contract (shown in Listing 1).

Such a contract interface can be stripped down to two simple methods, ie, a *registerJoinEui()* function and a *JoinEUI* getter. *registerJoinEui()* should have two properties, ie, forbidding registration of JoinEUI belonging to other networks and preventing future *JoinEUI*s to be known in advance. Registration of forged *JoinEUI*s would enable an attacker to duplicate captured upstream LoRaWAN packets from another join server to its node. While this does not pose a threat to the network itself, it causes unnecessary traffic on gateways and facilitates the collection of connection metadata associated with remote devices. We can enforce unique unforgeable applications identifiers by generating them in the smart contract. Since smart contracts only perform deterministic operations, we can build our identifiers from a constant seed or by using a recent block hash as a source of entropy. While block hashes are not strictly non-malleable, they can be used to prevent collisions. The JoinEUI space of 2^{64} becomes too small to neglect collisions for billions of devices (the birthday bound $\approx 5.4 \times 10^9$, see *birthday problem*). Therefore, we

```
contract JoinEuiRegistry is addressable {
    mapping (uint64 => address) public joinEuis;
    uint nonce = 1337;

    function registerJoinEui() public returns (uint64) {
        uint lastBlockHash = uint(blockhash(block.number - 1));
        uint64 joinEui = uint64(keccak256(
            abi.encodePacked(lastBlockHash, nonce)
        ));
        require(joinEuis[joinEui] == 0, "JoinEUI already registered");
        nonce = joinEui;
        joinEuis[joinEui] = msg.sender;
        emit JoinEuiRegistered(msg.sender, joinEui);
        return joinEui;
    }

    event JoinEuiRegistered(address indexed registrant, uint64 indexed joinEui);
}
```

Listing 1 *JoinEUI* registry smart contract in solidity

generate $JoinEUI_i = hash(block_N \| JoinEUI_{i-1})$ and check for unique JoinEUIs as a precaution. As a consequence, devices can only be provisonned after the registration of the application identifier.

3.2 | Activation procedure

End-device activation is required before participating in a LoRaWAN network. LoRaWAN supports two distinct activation methods, namely, Activation by Personalization (ABP) and Over-The-Air Activation (OTAA). ABP requires pre-provisioning of the encryption keys on the device and communication takes place without handshaking using a non-unique device address *DevAddr*. On the opposite, OTAA re-generates session keys during handshaking and devices identify themselves using a device identifier *DevEUI* and *JoinEUI*. Since we cannot unambiguously identify devices without giving up encryptions keys in ABP, we focus solely on OTAA. OTAA is preferred in most scenarios anyway due to its security advantages resulting from the regeneration of the session keys and its simplified network management.

The activation procedure is summarized in Figure 3. After the provisioning process (1), the end-device and the hNS must perform a handshake, ie, the activation, consisting of two messages: a join-request (2) sent by the end-device and a join-accept (5) sent back by the fNS. The join-request is sent in clear and consists of the *JoinEUI*, the *DevEUI*, and a nonce value. Since it is sent unencrypted, anyone intercepting the traffic can read its content, including the fNS. However, only entities possessing the *AppKey* can assert that the message has not been tempered with, using the message integrity code. In return, a join-accept (3) containing the device address (*DevAddr*) is sent encrypted using the *AppKey*. Since the fNS

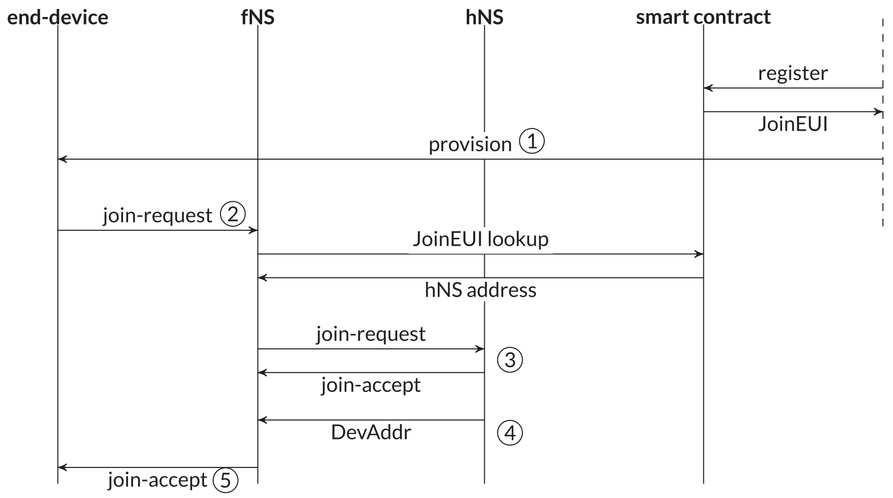

FIGURE 3 Activation procedure sequence

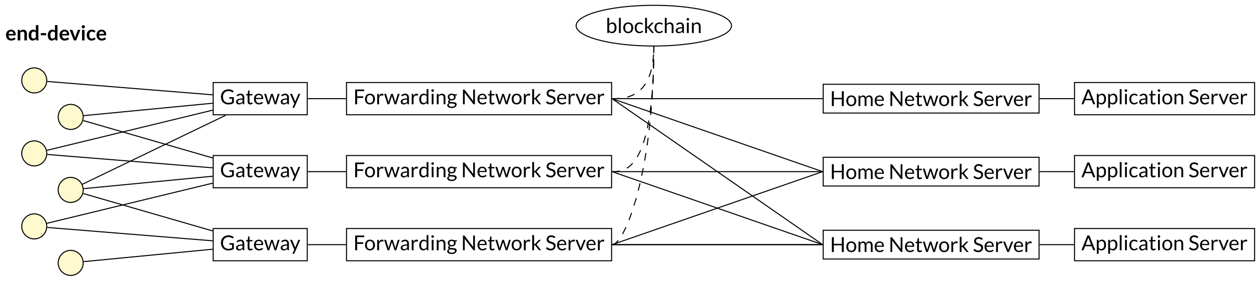

FIGURE 4 Decentralized LoRaWAN infrastructure

TABLE 2 Gas consumption from the *JoinEuiRegisty* smart contract

	execution cost	transaction cost		
operation	gas	gas	Microether	fiat
registerJoinEui()	28494	49766	99.53	$0.01194
setIpv4()[a]	20619	42275	84.55	$0.01015
setIpv4()[b]	5619	27275	54.55	$0.00655
setIpv6()[a]	20575	42999	86	$0.01032
setIpv6()[b]	5575	27999	56	$0.00672
getAddress()	0	0	0	0

[a]cost when the IP is first set. When the IP is changed on subsequant calls, the execution cost is decreased by 15 000 gas units as the SSTORE operation consumes more when the storage value is set to non-zero from zero.
[b]cost when the IP is updated.

cannot match a *JoinEUI* with a *DevAddr* without the *AppKey*, it could be sent out-of-band (4). Knowledge of *DevAddr* is required since it is sent alongside data messages and is used for routing.

3.3 | Forwarding scheme

To create a decentralized network such as the one illustrated in Figure 4, a stateful fNS (according to the definition from LoRaWAN Backend Interfaces[7]) dispatches LoRaWAN packets according to a set of simple rules. No message is forwarded in any direction until a session is created. Sessions are initiated from a join-request message, which is unencrypted and contains a *JoinEUI*. When a join-request is received by the fNS, its *JoinEUI* is mapped to an hNS (or its Join Server) address using the smart contract described in Section 3.1. This happens transparently since the router is connected to a local blockchain node. The join-request message is then forwarded to the hNS in expectation of a join-accept message. Since a join-accept message is fully encrypted (except for its header, we can assert that the message is a join-accept), the fNS can either (1) require a *DevAddr* along the join-accept or (2) broadcast subsequent messages to every hNS that can match the *DevAddr*.

3.4 | Testing and evaluation

This section describes our implementation of the stateful fNS presented in Section 3.3. This fNS consists of a Python forwarding server and a Solidity smart contract (as described in Section 3.1) running on the Ethereum blockchain. This implementation is compatible with LoRaWAN 1.0 devices and network servers. When receiving a join-request message, the fNS resolves the *JoinEUI* to an IPv4 or IPv6 address and forwards subsequent messages to it. The resolving process requires interaction with an Ethereum node. We compiled the contract and computed the gas required to call its functions. The consumed gas determines the fees paid by the users, which is highly dependent on the market price. The results are aggregated in Table 2 using a gas price of 2 Gwei and the exchange price as observed on January 17, 2019. Resolving a *JoinEUI* does not consume any gas as it is a view function, ie, it does not modify the state of the contract. To minimize the impact on the existing LoRaWAN software ecosystem, the proposed fNS accepts connections from a common LoRa packet forwarder, ie, "a program running on the host of a LoRa gateway that forwards RF packets received by the concentrator to a server through a UDP/IP link, and emits RF packets that are sent by the server."* Moreover, the fNS can run on the same host as the concentrator and acts as a client to the hNS. It does not therefore require an

* https://github.com/Lora-net/packet_forwarder

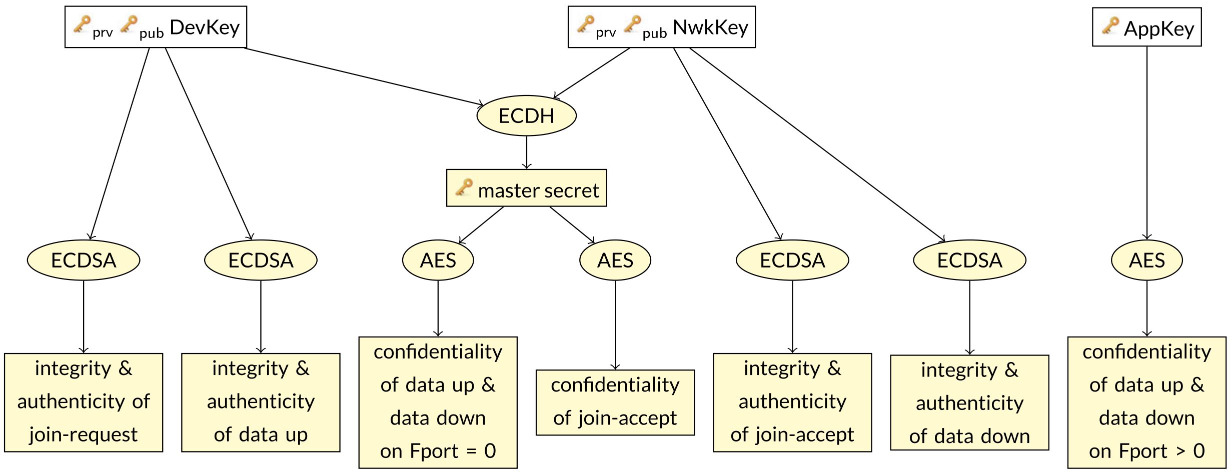
prv pub DevKey
prv pub NwkKey
AppKey
ECDH
master secret
ECDSA
ECDSA
AES
AES
ECDSA
ECDSA
AES
integrity & authenticity of join-request
integrity & authenticity of data up
confidentiality of data up & data down on Fport = 0
confidentiality of join-accept
integrity & authenticity of join-accept
integrity & authenticity of data down
confidentiality of data up & data down on Fport > 0

FIGURE 5 Proposed key derivation scheme

always-open listening address. Note that there are existing open-source application servers embedding a network server such as loraserver[†] or lorawan-server[‡] for small-scale deployments. Our project is freely available on GitHub.[§]

Since LoRaWAN 1.1 was recently released with no off-the shelf hardware or software stacks available at the time of writing, the prototype is limited to LoRaWAN 1.0. LoRaWAN 1.1 brings active roaming capabilities and a refined security model. In particular, the AppSKey is no longer derived from the same root network key (*NwkKey*). Moreover, a new set of keys are derived from the network key, which enables partial delegation of network management without the withdrawal of the complete network key. For instance, we can check the join-accept integrity by sharing the join session integrity key (*JSIntKey*). However, the security is still based exclusively on symmetric message integrity check and encryption. Sharing message integrity keys in an open network environment would defeat its purpose as it would allow anyone to create valid messages.

4 | LORAWAN ON STEROIDS: USING DIGITAL SIGNATURES

As previously stated, the LoRaWAN security model is built on symmetric cryptography. The message integrity and authenticity is checked using a message authentication code (MAC). In order to authenticate LoRa messages on public networks, we propose a LoRaWAN variant built on asymmetric cryptography. Adding digital signatures to LPWAN messages creates an opportunity for a whole new decentralized software ecosystem built upon smart contracts. As the message signer can be verified from a smart contract, it becomes possible to enforce fair use of the network, as proposed in BcWAN,[5] by rewarding gateway owners using digital tokens. Moreover, it becomes possible to create context-specific applications running on a blockchain. Typical use-cases include monetized applications and supply-chain accounting, but possible use is much broader according to a recent review on the use of blockchain for the IoT.[10]

To evaluate such a proposal on LoRa networks, we will examine the changes that are necessary. To keep the comparison fair, we keep an almost identical feature set to LoRaWAN and only upgrade the security model.

4.1 | Key derivation scheme

To ensure non-repudiation (in addition to integrity and authenticity) of a packet, forwarders need to authenticate end-devices or hNS (depending on the direction of the message) using digital signatures. We define a new security model based on two elliptic-curve key pairs ($\{DevKey_{prv}, DevKey_{pub}\}$, $\{NwkKey_{prv}, NwkKey_{pub}\}$) and an additional symmetric key (*AppKey*). Keys and their usage are depicted in Figure 5. In this model, *DevKey* and *NwkKey* ensure security on the MAC layer, while the AppKey ensures confidentiality of upper layer(s). $DevKey_{prv}$ is stored on end-devices and $NwkKey_{prv}$ is stored on hNSs. *AppKey* is provisionned on both the end-devices and the application servers; its role is identical in LoRaWAN 1.1 and it will not be further discussed. Confidentiality of messages is ensured by AES encryption from a shared master secret computed using elliptic-curve Diffie-Hellman (ECDH).

We evaluated the possibility of adding forward secrecy to the MAC-layer security. Perfect forward secrecy could be achieved using elliptic-curve Diffie-Hellman ephemeral (ECDHE) as performed by Sciancalepore et al[13] in a wireless network. This requires a one round-trip handshake and thus could be embedded in join-request and join-accept messages. However, as application data encryption does not rely on MAC-layer encryption, we judged that the overhead induced by the handshake was generally not worthwhile.

[†]https://github.com/brocaar/loraserver
[‡]https://github.com/gotthardp/lorawan-server
[§]https://github.com/DurandA/lora-peer

4.2 | Evaluation

This section discusses the pitfalls of public key cryptography for constrained devices (especially those running on battery) and evaluates whether the use of digital signatures is generally suited for LoRa-based devices. There are a few drawbacks related to the use of public key cryptography, which are of big concern in the context of constrained devices.

- Most operations are computationally more intensive.
- It requires longer keys for the same security level.
- Digital signatures are typically longer than MACs.

In the LPWAN context, these pitfalls translate into higher power consumption and decreased reliability. Higher power usage is due to longer time spent on cryptographic operations and on transmission, a direct consequence of increased message sizes due to the digital signature. Decreased reliability is also due to longer messages as they are more likely to collide to with other messages or to be corrupted during transmission.[14]

In the security model described in Section 4.1, ECDSA signature or verification is performed on every message by end-devices. On the opposite, other asymmetric operations, ie, DevKey generation and master secret derivation, can be done in advance, possibly outside of the device. We will characterize the impact of digital signatures on message size, power usage, and required resources on a modern low power microcontroller. Furthermore, we will experiment with a hardware security module to see if it results in a significant drop in power consumption.

4.2.1 | Message size

In LoRaWAN, the integrity and authenticity of messages is ensured by a block-cipher-based MAC (CMAC), a type of MAC that serves the same purpose as a hash-based MAC, but is built on a cipher function rather than a hash algorithm. The message integrity code (MIC) is defined as follows (adapted from LoRaWAN specification[6p26]):

$$cmac = aes128\_cmac(Key, B_n|msg) \tag{1}$$

$$MIC = cmac[0..3]. \tag{2}$$

The MIC is thus 4 bytes long, as it is the first 4 bytes of the calculated CMAC. As specified in Figure 6, the MIC of the *PHYPayload* is replaced by a digital signature. More formally, we define it as follows:

$$signature = ecdsa(DevKey_{prv}, msg). \tag{3}$$

This obviously results in an increased message size. ECDSA typically have signatures of $2n = |R| + |S|$, where n is the size (in bits) of the key and R and S are the components of the signature. While ECDSA is probably the most widely used signing scheme, there are other ones, some of them purposely built for small signatures sizes such as HFEV- and Gui.[15] In these cases, short signatures are achieved at the expense of large key sizes. Fortunately, complete public keys can be distributed using a public key infrastructure, possibly running on a blockchain, and we can use a hashed version of them as an identifier. Table 3 quantifies the PHYPayload size with different signature schemes. In this table, the message size is calculated by replacing the MIC with a signature and using $DevAddr = hmac[0..3]$ where $hmac = sha256(DevKey_{pub})$.

It is clear from the table that the increased message footprint when using digital signatures is significant. While evaluating the LoRaWAN scalability requires a complex model with many parameters,[14] we can infer the consequences of longer messages when applying duty-cycles. In Europe, the ETSI regulations dictates a 1% maximum duty-cycle on most 868 Mhz subbands. Using AN1200.13[16] formulas, we calculate the minimum time between subsequent packets starts in Table 4. Compared to LoRaWAN, the time between packets almost doubles when using ECDSA. In other terms, the bandwith is reduced by more than half, and applications requiring frequent updates are limited to fewer messages. Gui partially mitigates these issues and is probably acceptable for a lot of applications.

4.2.2 | Signing operations on constrained devices

As previously stated, performing asymmetric cryptography is more demanding. In the context of LPWAN, this translates to more time spent on cryptography, which, in turn, consumes more energy. To quantify how much energy is required to perform a signature, we ran ECDSA

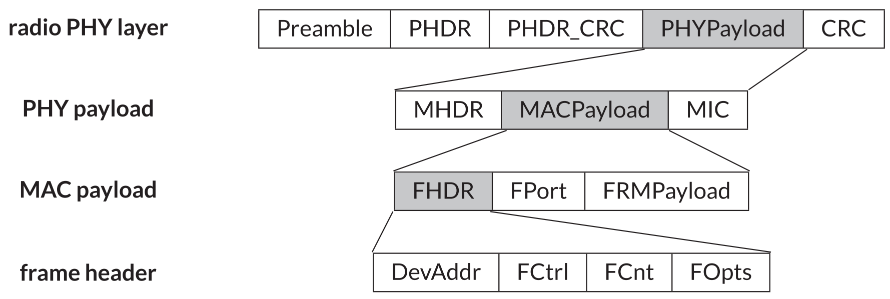

FIGURE 6 LoRaWAN MAC layer

TABLE 3 Message sizes without FOpts (in bytes)

	data message[a]
LoRaWAN	5+|M| bytes
ECDSA-160	41+|M| bytes
ECDSA-192	49+|M| bytes
ECDSA-256	65+|M| bytes
Gui[b]	17+|M| bytes

[a]confirmed and unconfirmed data up and data down message all have the same message structure.
[b]using scheme GUI(GF(2),95,9,5,5,3) from Section 8 in the work of Mohamed and Petzoldt.[15]

TABLE 4 Minimum time between data packets for 1% duty-cycle

payload (bytes)	LoRaWAN	ECDSA-192	Gui[a]
8 w/SF6	3.0848 s	6.9248 s	4.1088 s
8 w/SF9	18.5344 s	39.0144 s	24.6784 s
8 w/SF12	131.8912 s	246.5792 s	164.6592 s
20 w/SF6	4.1088 s	7.9488 s	5.1328 s
20 w/SF9	24.6784 s	45.1584 s	30.8224 s
20 w/SF12	164.6592 s	279.3472 s	197.4272 s

[a]using the same scheme as in Table 3.

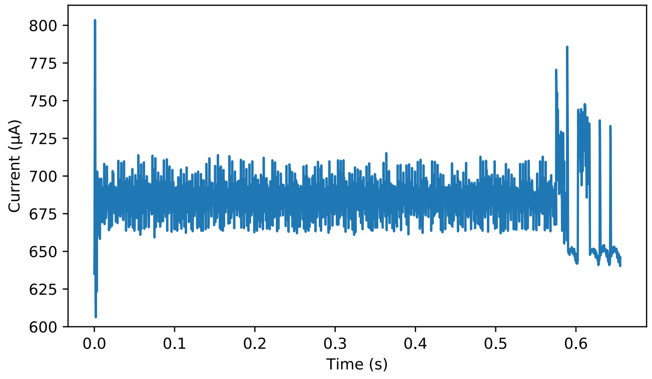

FIGURE 7 Current draw during ECDSA signing on SAML11

using (1) an optimized software implementation on a recent low-power microcontroller and (2) a discrete hardware cryptographic accelerator. To the best of our knowledge, there is no similar public evaluation on cutting-edge low power hardware. To evaluate software operations, we use an ultra low power SAML11 ARM Cortex-M23 microcontroller from Microchip Technology running at 8 Mhz, with the micro-ecc library.[¶] For the discrete cryptographic accelerator, we evaluate the ATECC508A Crypto Authentication device from the same manufacturer. A discrete cryptographic accelerator provides an easy upgrade path to existing hardware. In addition, we compare these operations to the CMAC function performed on 20 bytes. In both cases, we measure the current draw from the component that performs the operation (the microcontroller or the secure element) using a 3.3V supply, excluding other components of the device such as the LoRa radio, the power regulator, or LEDs. Figure 7 illustrates a single measurement of the electrical current during the *ecdsa_sign*() operation (using the secp160r1 curve) on the SAML11. The time window of this figure is cut to fit the operation time span since we are only interested in the required energy during the signature. The energy is then calculated from the electric charge at 3.3V. We performed such a measurement for each operation and synthesized the results in Table 5. As a comparison, the *aes*128_*cmac*() operation,[#] as performed to calculate the MIC, is also measured. From these measurements, we note a thousand-fold increase for both time and energy when performing ECDSA operations compared to CMAC. While this increase seems huge, the measured energy means that it is possible to perform more than one million ECDSA-160 signatures or verifications, excluding other device operations, on a CR2032 button cell battery. We also note a significant speed boost as well as the drop of energy consumption while using

[¶]https://github.com/kmackay/micro-ecc
[#] as implemented in Appendix A. Test Code of RFC-4493 document using AES hardware accelerator embedded in SAML11.

TABLE 5 Energy budget for cryptographic operations

	SAML11		ATECC508A	
operation	time	energy	time	energy
sign (secp160r1)	0.656 s	1.478 mJ	-	-
verify (secp160r1)	0.594 s	1.348 mJ	-	-
sign (secp256r1)	1.966 s	4.586 mJ	0.769 s	2.285 mJ
verify (secp256r1)	1.846 s	4.300 mJ	0.544 s	1.635 mJ
AES128_CMAC	0.575 ms	1.088 μJ	-	-

dedicated hardware (ATECC508A). Unfortunately, this device is only compatible with the secp256r1 curve so there is no comparison for smaller keys. Fast signing and verification could be required for time-sensitive applications, but LoRaWAN is not suited for real-time applications.[14] When taking these results into consideration, the penalty induced by digital signature seems acceptable in most scenarios.

5 | CONCLUSION AND FUTURE WORK

We have demonstrated a fully decentralized LoRaWAN architecture based on passive roaming techniques and discussed the limitations of such a system without non-repudiation. This paper focuses on crowd-sourced networks, but commercial operators could also benefit from this decentralized architecture. This would allow Join Server resolving without relying on the LoRa Alliance or having to build custom mechanisms to share *JoinEUI*s with partners.

We then proposed to change the security model, thus breaking compatibility, with the addition of digital signatures. This feature could enhance fairness and enable decentralized applications. We concluded that the overhead induced by digital signatures is acceptable for most applications. However, building decentralized systems on top of LPWAN requires further research on scalability, especially on public blockchains, due to the large amount of data produced by sensor networks.

Finally, we did not explore the possibility of using zero-knowledge proofs to prove the authenticity of vanilla LoRaWAN messages. In particular, we believe that it is possible to use succinct non-interactive zero-knowledge proofs (zk-SNARKs) to convince participants that a hNS possesses a valid *AppKey* matching the device address. However, best-in-class AES circuits have too many constraints[17] for a complete proof to be generated within seconds.

ORCID

Arnaud Durand https://orcid.org/0000-0002-7629-2121

REFERENCES

1. Centenaro M, Vangelista L, Zanella A, Zorzi M. Long-range communications in unlicensed bands: the rising stars in the IoT and smart city scenarios. *IEEE Wirel Commun*. 2016;23(5):60-67.
2. The Things Network. https://www.thethingsnetwork.org/. Accessed February 28, 2019.
3. Lin J, Shen Z, Miao C, Liu S. Using blockchain to build trusted LoRaWAN sharing server. *Int J Crowd Sci*. 2017;1(3):270-280.
4. Durand A, Gremaud P, Pasquier J. Resilient, crowd-sourced LPWAN infrastructure using blockchain. In: Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems; 2018; Munich, Germany.
5. Bezahaf M, Cathelain G, Ducrocq T. BcWAN: a federated low-power WAN for the Internet of Things (industry track). In: Proceedings of the 19th International Middleware Conference Industry; 2018; Rennes, France.
6. Sornin N, Yegin A. *LoRaWAN 1.1 Specification*. Technical Report. Fremont, CA: LoRa Alliance; 2017. www.lora-alliance.org/for-developers
7. Sornin N. *LoRaWAN Backend Interfaces 1.0 Specification*. Technical Report. Fremont, CA: LoRa Alliance; 2017. www.lora-alliance.org/for-developers
8. Nakamoto S. Bitcoin: a peer-to-peer electronic cash system. 2008.
9. Szabo N. Formalizing and securing relationships on public networks. *First Monday*. 1997;2(9).
10. Fernández-Caramés TM, Fraga-Lamas P. A review on the use of blockchain for the Internet of Things. *IEEE Access*. 2018;6:32979-33001.
11. Christidis K, Devetsikiotis M. Blockchains and smart contracts for the Internet of Things. *IEEE Access*. 2016;4:2292-2303.
12. Bahga A, Madisetti VK. Blockchain platform for industrial Internet of Things. *J Softw Eng Appl*. 2016;9(10):533.
13. Sciancalepore S, Piro G, Boggia G, Bianchi G. Public key authentication and key agreement in IoT devices with minimal airtime consumption. *IEEE Embed Syst Lett*. 2017;9(1):1-4.
14. Adelantado F, Vilajosana X, Tuset-Peiro P, Martinez B, Melia-Segui J, Watteyne T. Understanding the limits of LoRaWAN. *IEEE Commun Mag*. 2017;55(9):34-40.
15. Mohamed MSE, Petzoldt A. The shortest signatures ever. Paper presented at: International Conference in Cryptology in India; 2016; Kolkata, India.

16. *SX1272/3/6/7/8: LoRa Modem*. Technical Report No. AN1200.13. Camarillo, CA: Semtech Corporation; 2012. https://www.semtech.com/uploads/documents/LoraDesignGuide_STD.pdf
17. Kosba A, Papamanthou C, Shi E. xJsnark: a framework for efficient verifiable computation. Paper presented at: 2018 IEEE Symposium on Security and Privacy (SP); 2018; San Francisco, CA. https://doi.org/10.1109/SP.2018.00018

How to cite this article: Durand A, Gremaud P, Pasquier J. Decentralized LPWAN infrastructure using blockchain and digital signatures. *Concurrency Computat Pract Exper*. 2019;e5352. https://doi.org/10.1002/cpe.5352