REALISE-IoT: RISC-V Based Efficient and Lightweight Public-key System for IoT Applications

Gaoyu Mao, Yao Liu*, *Member, IEEE,* Wangchen Dai, Guangyan Li, Zhewen Zhang, Alan H.F. Lam, *Member, IEEE,* Ray C.C. Cheung, *Senior Member, IEEE*

***Abstract*—LoRa is a promising choice for deploying an IoT network due to its lightweight feature and the extensive support by LoRa Alliance. However, as a fundamental part of LoRa, the typical LoRaWAN protocol confronts severe security challenges because it insecurely utilizes AES-128 to support the low-cost feature. In this paper, we propose a systematic solution that is compatible with LoRaWAN for IoT applications. We extend the standard LoRaWAN protocol with public-key infrastructures. Public-key features like Key exchange and authentication are supported by lightweight hardware implementations of SHA-2, ECDH, EdDSA, and TRNG. A lightweight RISC-V processor with a security coprocessor is implemented and verified using FPGA technology. The security protocol and the prototype hardware system are validated and evaluated on practical applications from our industrial partner. The prototyped development board consumes a static power of 0.116 W and a dynamic power of 0.206 W. The proposed system can achieve a 5.6x-144.7x speed up and reduce memory usage by 2.4x-12.3x for security computations.**

***Index Terms*—Internet of Things, LoRa network, LoRaWAN, RISC-V, Public-key cryptography, Lightweight cryptography**

I. Introduction

The Internet of Things (IoT) is one of the key technologies for the next generation of industrial revolution [1]. It can get linked to everything, hence getting popular day by day [2]. Wireless communication technology acts as the bridge between data collection and control message delivery, facilitating IoT expansions. Low-power wide-area networks (LPWANs) become the wireless communication backbone for long-range interconnection between diversified IoT devices [3]. LoRa, a popular radio modulation technology licensed by Semtech Corporation, provides a long-range communication approach with the chirp spread spectrum technique (CSS) [4]. LoRa-based communication systems can be deployed in various scenarios, such as suburban environments [5], remote field sites [6], multi-floor buildings [7], et al. While there are other networks, such as Wi-Fi and Bluetooth, LoRa is a better choice for long-range communication and low-power applications[8]. However, the LoRa only defines the standards of the physical layer. To support the increasing demand for IoT connectivity, LoRaWAN [9] defines the upper layers and communication protocol of LoRa-based networks. LoRaWAN connects and links the LoRa signal to applications. Lora and LoRaWAN define the network protocol in the LPWANs family to connect battery-operated devices to the internet wirelessly. LoRa devices and LoRaWAN standards provide flexibility in many industrial use cases and dominate the market.

For the consideration of lightweight features, most IoT applications based on LPWAN technologies are vulnerable to hostile attackers. For example, despite the satisfactory security level of AES-128 encryption, the use of AES-128 in the standard LoRaWAN protocol introduces potential weaknesses as a price of the lightweight feature. To increase security, many enhanced protocols for LoRaWAN have been proposed. Naoui et al. [10] enhanced the security of LoRaWAN by applying proxy nodes and a reputation system to alleviate the computation tasks. Han et al. [11] proposed a lightweight key management scheme based on the Rabbit cipher embedded in a two-step Key Derivation Function (KDF). To overcome the challenge of secure key generation at long distances and low data rates, Xu et al. [12] proposed a compressive sensing-based reconciliation framework combined with several signal processing techniques to achieve secure key generation. To further support indoor-to-outdoor scenarios and to reduce the correlation between channel measurements, Junejo et al. [13] presented a shared secret key generation scheme with several processing techniques and achieved a low correlation value, low key disagreement rates, and high key generation rates. However, these solutions lack strong public-key cryptography support due to the limited computing resources and low-power consumption requirements of LoRa devices [14], [15].

IoT applications generate large amounts of data and require complex computations. Edge computing solutions place IoT devices close to data sources to facilitate real-time processing. With built-in processors, IoT devices can accommodate advanced computing requirements. Current options for developing IoT processors include ARM, x86, and RISC-V. Jung

This work is supported by the Hong Kong Innovation and Technology Commission (ITF Seed Fund ITS/216/19), City University of Hong Kong (Project Grant No. 9440242 and 9678187), the Hong Kong Innovation and Technology Commission (InnoHK Project CIMDA), and National Natural Science Foundation of China (No. 62002239). (Corresponding author: Yao Liu.)

Gaoyu Mao, Guangyan Li, Zhewen Zhang, Alan H.F. Lam, and Ray C.C. Cheung are with the Department of Electrical Engineering, City University of Hong Kong, Kowloon Tong, Hong Kong SAR, China (e-mail: gaoyumao3-c@my.cityu.edu.hk; guangyali5-c@my.cityu.edu.hk; zhewzhang3-c@my.cityu.edu.hk; hiuflam@cityu.edu.hk; r.cheung@cityu.edu.hk).

Yao Liu is with the School of Microelectronics Science and Technology, Sun Yat-Sen University, Zhuhai, China (e-mail: liuyao25@mail.sysu.edu.cn).

Wangchen Dai is with the Zhejiang Lab, Hangzhou, China (e-mail: w.dai@my.cityu.edu.hk).

et al. [16] proposed a secure platform model for low-end IoT devices based on ARM platform security architecture. It consisted of system security services and application security services and provided APIs for easy and fast development. Considering the integration of RISC-V and IoT devices, Amor et al. [17] extended the RISC-V ISA to achieve improved support for ultra-low power wireless communication. Taheri et al. [18] extended the RI5CY core with ISA extension for Hyperdimensional computing. It achieved 7.48× speed up and 7.22× energy efficiency. However, no additional security features at the hardware level were included in these works, which failed to enhance the security mechanism.

RISC-V is an open-source Instruction Set Architecture (ISA) based on the reduced instruction set computer (RISC) principles [19]. It consists of three basic instruction sets and six extended instruction sets. With high flexibility, RISC-V can be extended with additional instructions for specific applications. The RISC-V architecture allows open-source processor designs for both FPGA and ASICs. The processor can be customized for specific IoT applications based on the existing RISC-V design. PULPino is an open-source platform based on the RISC-V architecture. It is based on 32-bit RISC-V cores developed at ETH Zurich [20], [21]. It can be configured to use either the RISCY or the zero-riscy core. The RISCY is a single-issue core with four pipeline stages, while the zero-riscy is an in-order, single-issue core with two pipeline stages. The zero-riscy core is designed to target low power and lower area constraints. It can be configured to the lightweight version with only 16 General-purpose registers (GPRs).

To enhance the security of the LoRaWAN protocol and make it practical for IoT applications, we propose a systematic solution for the LoRaWAN security communication system. The main contributions are summarized as follows:

- The security of the LoRaWAN protocol is enhanced with minimal cost. The enhanced protocol provides public-key features and is compatible with the LoRaWAN standard. The elliptic-curve cryptography algorithms are applied. Specifically, the X25519 algorithm is selected to enhance the key exchange process, while the Ed25519 algorithm is chosen to add the digital signature. The compatible protocol can gain support from LoRa Alliance and hence is practical to deploy in real applications.
- A systematic solution is proposed from network protocol to system architecture and hardware devices. The platform is developed based on PULPino, and multiple hardware modules are included for applications. For instance, the LoRa network interface is integrated for communication, a digital TRNG core acts as the root of trust, and a lightweight security coprocessor is designed for public-key cryptography computations. Furthermore, an efficient point multiplication architecture is designed, and an effective modular multiplication algorithm is explored for hardware implementation.
- The proposed system supports lightweight features for IoT applications. The compact point multiplication module contains only one multiplier, adder, and subtractor. The addition and subtraction operations are hidden in the multiplication pipeline cycles to make a compact timing schedule. The platform is implemented with reconfigurable logic on a low-cost FPGA and prototyped on a PCB board. The proposed system can achieve 5.6x-144.7x speed up and reduce memory usage by 2.4x-12.3x. The overall estimated static power is 0.116 W, and the dynamic power is 0.206 W.

The rest of this paper is organized as follows. Section II illustrates the background. Section III describes the enhanced security protocol, the system architecture for the LoRaWAN communication system, and efficient implementations for ECC. Section IV details the hardware architecture of the secure processor and security modules. Section V presents the evaluation results. Section VI concludes this paper.

II. Background

A. Security protocol for LoRaWAN

From the security perspective, a typical working process for a LoRaWAN device consists of four phases, as shown in Fig. 1.

- *Deployment*: the LoRa node is fabricated with the root key K_r and the unique device info I_d, including the AppEUI, DevEUI, etc. The network server needs to know K_r and partial information of I_d before the LoRa node is deployed so that the server can authenticate the identity of the LoRa device in the following phases.
- *Join*: the LoRa node does not bear a routable address in the network, and a gateway is needed to relay and reassemble the frames. The LoRa device must complete the join process before communicating with the server. The join message is a plaintext with device nonce N_d, affiliated with the Message Integrity Code (MIC) calculated with K_r. The server verifies the correctness of the join message by the pre-distributed information.
- *Acceptance*: the server needs to send a join-accept message to the LoRa node after the LoRa device is successfully authenticated. Application information I_a and application nonce N_a are added to the message, and the join message is encrypted by K_r together with the MIC. The LoRa device recovers the extra information N_a and I_a with K_r, and generates the session key K_s. Since the server knows all the information, the same session key can be generated in the server.
- *Communication*: the LoRa node and the server communicate with each other using K_s. Since a typical LoRaWAN frame cannot exceed 255 bytes, the LoRaWAN provides a bytewise encryption approach. Each byte in the payload is encrypted by XORing (exclusive-OR) an encryption block uniquely generated by the frame information and the position information of the block.

The details of the security protocol slightly change in LoRaWAN v1.1. More keys are involved, but the security protocol remains similar with no significant security enhancement [22]. Note that this protocol only applies to Over-The-Air Activation (OTAA) devices. As for Activation By Personalization (ABP) devices, the join and acceptance phases are skipped. The session key is directly stored inside the APB devices so that the session key remains unchanged

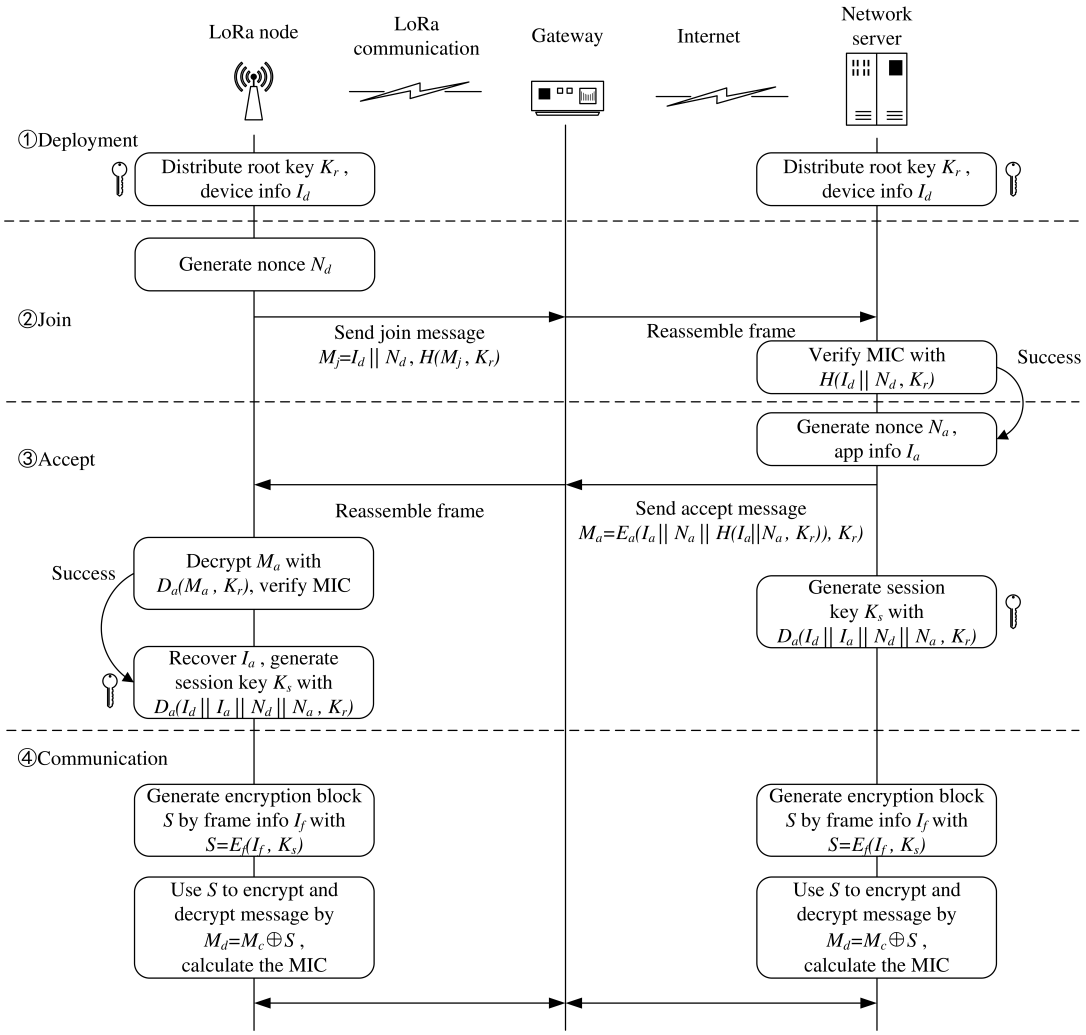


Fig. 1. Security protocol for LoRaWAN v1.0 with OTAA.

without redistribution [23]. However, the deployment scenarios of LoRa networks are usually in the wild without human management, such as outdoor pipelines, farmland, or deep mountains. Pre-deployed session keys are easily available in these scenarios, making it easy to steal or forge transmission information. Therefore, we only consider OTAA for secure wireless transmission in this paper.

The security protocol is delicately designed, aiming for lightweight applications. Only AES-128 encryption is needed in the LoRa node, and the decryption part is only necessary for the server. For the MIC generation function $H()$, AES-CMAC is applied. For the key generation process, $E_a()$ is the AES-128 decryption in the server, and the LoRa node only needs to perform the AES-128 encryption as the $D_a()$ process. Encryption block S is generated by AES-128 encryption $E_f()$, and the encryption process during the communication phase is essentially the Feistel cipher with S so that the encryption and decryption share the same processes.

The lightweight feature of the security protocol brings potential weaknesses to the whole system. The key generation scheme is based on the fixed root key and never updates. The system will be compromised once a third party obtains this key [24]. Aras et al. [23] launched an experiment on the Xignal mousetrap device and extracted the keys via the physical access of the UART interface. With keys and a custom LoRa device, they impersonated a LoRa mouse trap and sent the data pretending from it. Furthermore, LoRaWAN implements AES in counter mode. If the counter values repeat with the same key, the same keystream is used for encryption, voiding confidentiality [25]. Yang et al. [26] conducted a proof-of-concept experiment to evaluate the data recovery attack caused by the key stream reuse issue. The TTN Fair Access Policy somewhat reduces the possibility of data leakage by restricting the transmitted data volume in a fixed period to share the communication channel fairly [27]. Nevertheless, it is not a compulsory rule for all LoRa devices to follow.

B. Lightweight public-key cryptography

The public-key cryptographic systems use two key pairs: public and private keys. The private key is kept secret, while the public key can be distributed to the public without compromising security. Hence, public-key cryptography provides

a strong security guarantee and is ideal for security extensions. However, public-key cryptography has a relatively large size and low processing speed, which may not be suitable for LoRa applications. For example, the RSA algorithms use very large key sizes (e.g., $1024, 2048$, or 4096 bits), which require large hardware areas and expensive arithmetic calculations [28]. The popular lattice-based cryptography schemes are also relatively large in size. For example, the lattice-based signature CRYSTALS-Dilithium [29] has a public size of 1312 bytes and a signature size of 2420 bytes in Dilithium2. However, a typical LoRaWAN cannot exceed 255 bytes. If post-quantum cryptography is applied for LoRa applications, it is necessary to propose a more lightweight scheme with a smaller key and signature size and conduct a security analysis.

Elliptic-curve cryptography (ECC) achieves an equivalent security level with smaller key sizes and computation overhead than other public-key cryptography schemes. For example, to achieve a security level of 128 bits, the traditional RSA scheme requires 3072 bits operands while ECC only requires 256 bits. The security strength of ECC schemes depends on the underlying elliptic curves. Due to the concern about the potential security weakness of the curve recommended by the National Institute of Standards and Technology (NIST), the Montgomery curves, mainly Curve25519 [30], have drawn increasing attention. Curve25519 is faster, more secure, and simpler than other ECC curves; therefore, it has recently been recommended by IETF RFC 7748 and integrated into popular SSL/TLS libraries [31].

The Elliptic-curve Diffie–Hellman (ECDH) key exchange protocol can generate a shared secret key through insecure channels. The secure key exchange protocol can address the potential weakness in generating the symmetric AES key in LoRaWAN. The shared secure key generated by ECDH can be adopted as the subsequent AES key to enhance security strength. The Edwards-curve Digital Signature Algorithm (EdDSA) is a digital signature scheme that provides a layer of validation and security to messages through non-secure channels. The generated signature can be added to the LoRaWAN communication packet for message authentication. We select the X25519 for ECDH based on RFC 7748 [32] and the Ed25519 for EdDSA based on RFC 8032 [33].

III. Methodology

A. Enhanced security protocol with ECDH key exchange

The primary security deficiency of the existing LoRaWAN protocol lies in the fixed root key, which is used repeatedly to generate the session key for symmetric encryption between different end devices. To address this security problem, a two-level security protocol is proposed to provide an enhanced security guarantee for a general IoT system. The proposed protocol is decoupled with the underlying IoT protocol; thus, it is compatible with various IoT networking techniques. LoRaWAN is adopted in this work to demonstrate the working principle of the proposed protocol as presented in Figure 2. The proposed protocol enhances security with a two-level structure. Level 1 is the underlying IoT communication protocol, which is LoRaWAN in this work. After the

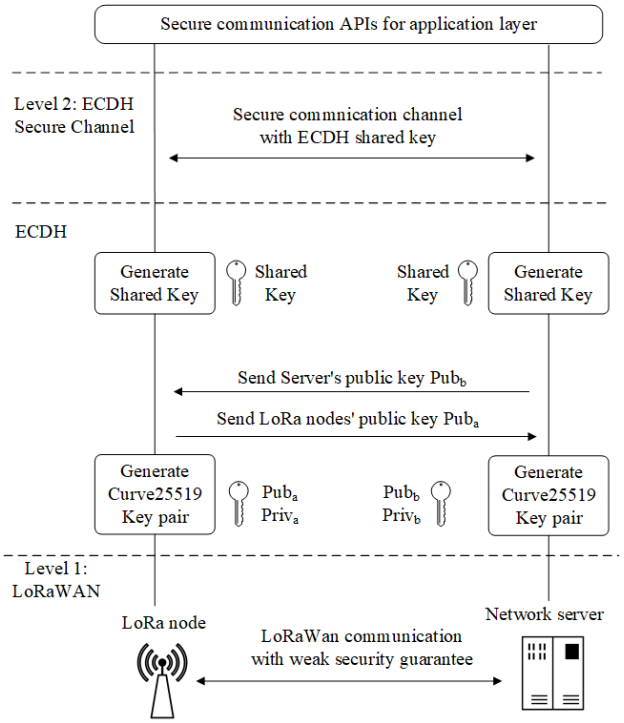


Fig. 2. The enhanced protocol with ECDH.

LoRaWAN communication channel is established, the LoRa nodes and servers will generate their own ECC key pairs with Curve25519. The key exchange algorithm is shown below, where $\mathbf{S_{DH}}()$ is scalar multiplication defined on an elliptic curve, and G is the base point for the chosen elliptic curve.

$$Pub_a = \mathbf{S_{DH}}(Priv_a, G) \quad Pub_b = \mathbf{S_{DH}}(Priv_b, G)$$
$$SK_{ab} = \mathbf{S_{DH}}(Priv_a, Pub_b) = \mathbf{S_{DH}}(Priv_b, Pub_a) = SK_{ba}$$

During ECDH, each end device only distributes its public key through the insecure communication channel and keeps its private key. ECDH guarantees the same secret keys pairs (SK_{ab}, SK_{ba}), as long as two parties (a, b) using the same base point G and have access to each other's public key (Pub_a, Pub_b). Meanwhile, a strong security level is guaranteed whenever each side protects the private key $(Priv_a, Priv_b)$ properly. A third party with the public key of both sides still needs to crack down the secret key with brute force. With the shared secret key, a secure communication channel is established. The application layer can assume the underlying communication layer is secure with the corresponding APIs. With this protocol, various applications with higher security requirements can be performed on IoT devices.

B. Enhanced security protocol with EdDSA signature

The standard LoRaWAN frame contains header information and payload messages, as shown in Figure 3. One security deficiency of the LoRaWAN frame is the lack of message authentication, so the message receiver is hard to identify the message sender without authentication. To address this problem, a digital signature can be attached to each frame. Since

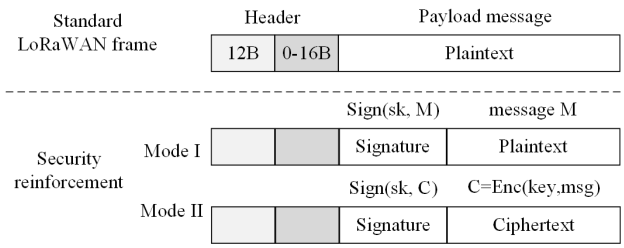


Fig. 3. The enhanced LoRaWAN frame with a digital signature.

the size of the EdDSA signature is only 512-bit (64 bytes), the generated signature can be put within each LoRaWAN frame. The security-enhanced LoRaWAN frame is shown in Figure 3. Two modes of usage scenarios are provided. The first is to send the plaintext message, and the signature is signed for the plaintext. While the other is first to encrypt the message to obtain a ciphertext, and then a signature is signed for the ciphertext.

The EdDSA algorithm is used to create the signature. The signature is generated in Curve25519, the same as ECDH. The signature algorithm is shown below. The $\mathbf{S_{DSA}}$ is two coordinates scalar multiplication in EdDSA. The $\mathbf{SHA_{512}}$ is the hash function of SHA-2. The input message is hashed with $\mathbf{SHA_{512}}$ to produce a 512-bit output. Two modulo operations (mod q) are required, where $q = 2^{255} - 19$. The signature is the concatenation of two intermediate results, R and s.

$$\begin{gathered} r = \mathbf{SHA_{512}}(Priv'_a[32:63] + msg) \mod q \\ R = \mathbf{S_{DSA}}(r, G) \\ h = \mathbf{SHA_{512}}(R + Pub'_a + msg) \\ s = (r + h \cdot Priv'_a[0:31]) \mod q \\ Signature = R||s \end{gathered}$$

The LoRa nodes create the public key Pub'_a and private key $Priv'_a$. The public key is sent to the LoRa server to verify the message. The Pub'_a and the $Priv'_a$ are stored in the node to sign the messages. This public key is distributed through insecure channels. A third party with the public key Pub'_a can only verify the message but cannot create a new signature without knowing the private key $Priv'_a$.

C. Lightweight and efficient implementation of ECC

Scalar multiplication is the most time-consuming function in ECDH (X25519) and EdDSA (Ed25519). The scalar multiplication involves the modular multiplication $p = a \cdot b \bmod q, q = 2^{255} - 19$. We apply the property of modular q and obtain an efficient modular multiplication algorithm, as shown in Algorithm 1. In Algorithm 1, the parameter w is 32 and s is 8. The (t, u, v) is a 96-bit register, and t, u, v are 32-bit. In the whole algorithm, the 256-bit multiplication is split into 32-bit multiplications. It first calculates $a \cdot b \bmod 2q$ (i.e., Step 1-18), then calculates $a \cdot b \bmod q$ (i.e., Step 19-25). For the coefficients with an index exceeding $s - 1$, each coefficient is multiplied by $38 \cdot 2^{-256}$ (i.e., Step 7-8). The final result is modular q instead of modular $2q$, so the last word contains $w-1$ bits instead of w bits (i.e., Step 15-16). At the end of Step 18, we get the value of $x + x' \cdot 2^{255}$, where $x \in [0, 2^{255} - 1]$ and x' is a small value with less than $2w$ bits. Besides, x is stored in p_i in radix-2^w form and x' in the register array (t, u, v). Steps 19-24 reduce the value of x' modulo n by computing $19 \cdot x'$ and add the result back to x. It is easy to see the sum of $x + 19x'$ may be greater than n but less than $2n$, and therefore a final conditional subtraction is required (i.e., Step 25).

Algorithm 1 Modular multiplication for modulus $2^{255} - 19$

Require: Modulus $2^{255} - 19$, operand $a, b < 2^{256} - 1$ are in radix-2^w form with $a_i, b_i \in [0, 2^w - 1]$, $i = 0, 1, \ldots, s-1$ and $sw = 256$.
Ensure: $p = a \cdot b \bmod 2^{255} - 19$.

1: $(t, u, v) \leftarrow 0$
2: **for** i from 0 to $s - 1$ **do**
3: **for** j from 0 to $s - 1$ **do**
4: **if** $j \leq i$ **then**
5: $(t, u, v) \leftarrow (t, u, v) + a_j \cdot b_{i-j}$
6: **else**
7: $(t, u, v) \leftarrow (t, u, v) + a_j \cdot b_{i-j+s}$
8: $(t, u, v) \leftarrow (t, u, v) \times 38$
9: **end if**
10: **end for**
11: **if** $i < s - 1$ **then**
12: $p_i \leftarrow v$
13: $v \leftarrow u$, $u \leftarrow t$, $t \leftarrow 0$
14: **else**
15: $p_{s-1} \leftarrow v \bmod 2^{w-1}$
16: $(t, u, v) \leftarrow (t, u, v) \gg (w - 1)$
17: **end if**
18: **end for**
19: $(t, u, v) \leftarrow (t, u, v) \times 19$
20: **for** i from 0 to $s - 1$ **do**
21: $(t, u, v) \leftarrow (t, u, v) + p_i$
22: $p_i \leftarrow v$
23: $v \leftarrow u$, $u \leftarrow t$, $t \leftarrow 0$
24: **end for**
25: **If** $p \geq n$ **return** $p \leftarrow p - n$**, else return** p

During the realization of Algorithm 1, the final conditional subtraction can process with the reduction of $x + x' \cdot 2^{255}$ and thus save a subtraction with long propagation. This can be done by computing $y = x + 19x'$ and $y' = x + 19x' + 19$ in parallel. If the MSB (the 256-th bit) of y' (denoted as y'_{msb}) is set, we assign $p = y' \bmod 2^{255}$ and output, otherwise $p = y$. The conclusion can be proved as follows. Since $y < y' < 2n$, there exist three cases: both y_{msb} and y'_{msb} are set; only y'_{msb} is set; and both are zero. For both y_{msb} and y'_{msb} are set, $y \geq 2^{255}$, note that $y < 2n$, we only need to subtract one n from y: $y - n = (x + 19x' + 19) \bmod (2^{255}) = y' \bmod 2^{255}$. For only y'_{msb} is set, then $n \leq y < 2^{255}$, we still need to subtract one n from y: $y - n = y + 19 - 2^{255}$, notice that $y + 19 = y' \geq 2^{255}$, so $y - n = y' \bmod 2^{255}$. Finally, for both y_{msb} and y'_{msb} are zero, then $y' < 2^{255}$ and $y = y' - 19 < 2^{255} - 19 = n$, no subtraction is needed.

The timing schedule of scalar multiplication for X25519 is shown in Table I. The u and k are 256-bit input variables. The capital letters (e.g., A), x_i, z_i, and k are used to store

TABLE I
TIMING SCHEDULE OF SCALAR MULTIPLICATION FOR X25519

State	Multiplier	Adder	Subtractor	Control
Initialization: $x_1 = u$, $x_2 = 1$, $z_2 = 0$, $x_3 = u$, $z_3 = 1$, $swap = 0$, $t = 254$				
Main Loop				
1		$A \leftarrow x_2 + z_2$	$B \leftarrow x_2 - z_2$	$k_t \leftarrow (k \gg t) \wedge 1$, $swap \leftarrow swap \oplus k_t$ $(x_2, x_3) =$ MUX_2X2$(x_3, x_2 : swap)$ $(z_2, z_3) =$ MUX_2X2$(z_3, z_2 : swap)$
2	$E \leftarrow A \times A$	$C \leftarrow x_3 + z_3$	$D \leftarrow x_3 - z_3$	$swap \leftarrow k_t$
3	$F \leftarrow B \times B$			
4	$A \leftarrow A \times D$			
5	$B \leftarrow B \times C$	$C \leftarrow C + C$, $C \leftarrow C + D$, $C \leftarrow C + C$, $C \leftarrow C + D$ $C \leftarrow C + C$, $C \leftarrow C + C$, $C \leftarrow C + D$, $(29C)$	$C, D \leftarrow E - F$	
6	$x_2 \leftarrow E \times F$	$A \leftarrow A + B$ $C \leftarrow C + C$, $C \leftarrow C + D$, $C \leftarrow C + C$, $C \leftarrow C + C$ $C \leftarrow C + D$, $C \leftarrow C + C$, $C \leftarrow C + D$, $(475C)$	$A \leftarrow A - B$	
7	$B \leftarrow B \times B$	$C \leftarrow C + C$, $C \leftarrow C + C$, $C \leftarrow C + D$, $C \leftarrow C + C$ $C \leftarrow C + C$, $(7604C)$		
8	$z_3 \leftarrow B \times x_1$	$C \leftarrow C + C$, $C \leftarrow C + C$, $C \leftarrow C + C$, $C \leftarrow C + C$ $C \leftarrow C + D$, $(121665C)$, $C \leftarrow E + C$		
9	$z_2 \leftarrow C \times D$			
10	$x_3 \leftarrow A \times A$			$t \leftarrow t - 1$
Post Computation				
	$z_2 \leftarrow z_2^{p-2}$			$(x_2, x_3) =$ MUX_2X2$(x_3, x_2 : swap)$ $(z_2, z_3) =$ MUX_2X2$(z_3, z_2 : swap)$
	$x_2 \leftarrow x_2 \times z_2$			

the computation results. The MUX_2X2 function is to swap two inputs based on the $swap$ signal. The main computation is performed in the main loop. The loop is controlled starting from $t = 254$ to $t = 0$. The computation is conducted with one multiplier, adder, and subtractor. The timing schedule is arranged mainly based on the multiplication operation. Addition and subtraction are hidden in the multiplication cycle to achieve compact scheduling. The computation of $121665 \cdot c$ is split into several addition operations (State 5-8).

In the post computation of Table I, the inverse of z_2 is computed by z_2^{p-2}, where $p = 2^{255} - 19$. This exponentiation can be computed by 254 squaring and 11 multiplications. The addition chain of power is described as follows: $1 \rightarrow 2 \rightarrow 4 \rightarrow 8 \rightarrow \mathbf{9} \rightarrow \mathbf{11} \rightarrow 22 \rightarrow \mathbf{(2^5 - 1)} \rightarrow (2^6 - 2) \rightarrow \cdots \rightarrow (2^{10} - 2^5) \rightarrow \mathbf{(2^{10} - 1)} \rightarrow (2^{11} - 2) \rightarrow \cdots \rightarrow (2^{20} - 2^{10}) \rightarrow \mathbf{(2^{20} - 1)} \rightarrow (2^{21} - 2) \rightarrow \cdots \rightarrow (2^{40} - 2^{20}) \rightarrow \mathbf{(2^{40} - 1)} \rightarrow (2^{41} - 2) \rightarrow \cdots \rightarrow (2^{50} - 2^{10}) \rightarrow \mathbf{(2^{50} - 1)} \rightarrow (2^{51} - 2) \rightarrow \cdots \rightarrow (2^{100} - 2^{50}) \rightarrow\rightarrow \mathbf{(2^{100} - 1)} \rightarrow (2^{101} - 2) \rightarrow \cdots \rightarrow (2^{200} - 2^{100}) \rightarrow \mathbf{(2^{200} - 1)} \rightarrow (2^{201} - 2) \rightarrow \cdots \rightarrow (2^{250} - 2^{50}) \rightarrow \mathbf{(2^{250} - 1)} \rightarrow (2^{251} - 2) \rightarrow \cdots \rightarrow (2^{255} - 2^5) \rightarrow \mathbf{(2^{255} - 21)}$.

The timing schedule of scalar multiplication for Ed25519 is shown in Table II. The design principle is similar to X25519 shown in Table I. The P and k are input variables. The MUX function selects two inputs based on the k_t signal. The main loop starts from $t = 255$ to $t = 0$. Similar to X25519, only one multiplier, adder, and subtractor are used in the computation. Meanwhile, the modulo q is the same in X2559 and Ed25519. Therefore, a unified hardware architecture can be designed to accommodate the scalar multiplications for X25519 and Ed25519. Both multiplications share the same computation unit (multiplier, adder, and subtractor), while the control logic differs in the two functions. In summary, the unified architecture can achieve a more compact design. The cycles of addition and subtraction are hidden in the multiplication computation to achieve compact scheduling.

D. Systematic solution for LoRaWAN communication

The IoT platforms contain software, hardware, memory, sensors, network, user interfaces, etc. Integrating various components helps unlock IoT systems' full potential and build complete end-to-end IoT solutions. For a proper integration to occur, it requires a unifying platform. FPGAs contain programmable logic blocks, and reconfigurable interconnects. The logic blocks can perform complex functions and wire together through interconnections. The reconfigurable hardware feature of FPGA allows for providing customized IoT solutions without any physical hardware modifications. Moreover, to make

TABLE II
TIMING SCHEDULE OF SCALAR MULTIPLICATION FOR ED25519

State	Multiplier	Adder	Subtractor	Control
Initialization: $(X1, Y1) = (P_x, P_y)$, $(X2, Y2, Z2, T2) = (0, 1, 1, 0)$, $t = 255$				
$F = 2d = 741914118693388786862761670175091303790842277596864380327775710661718805671 10$				
Pre Computation				
	$F \leftarrow y_1 \times F$			
	$F \leftarrow x_1 \times F$			
Main Loop				
1	$A \leftarrow X2 \times X2$			$k_t \leftarrow (k \gg t) \wedge 1$,
2	$B \leftarrow Y2 \times Y2$	$C \leftarrow X2 + Y2$		
3	$A \leftarrow C \times C$	$C \leftarrow A + B$	$D \leftarrow A - B$	
4	$A \leftarrow Z2 \times Z2$		$B \leftarrow C - A$	
5	$Y2 \leftarrow C \times D$	$A \leftarrow A + A$		
6	$T2 \leftarrow C \times B$	$A \leftarrow A + D$		
7	$X2 \leftarrow A \times B$			
8	$Z2 \leftarrow A \times D$	$A \leftarrow Y2 + X2$ $C \leftarrow Y1 + X1$	$B \leftarrow Y1 - X1$ $D \leftarrow Y2 - X2$	
9	$B \leftarrow B \times D$			
10	$A \leftarrow A \times C$	$C \leftarrow Z2 + Z2$		
11	$D \leftarrow F \times T2$	$A \leftarrow A + B$	$B \leftarrow A - B$	
12	$TX \leftarrow A \times B$	$C \leftarrow C + D$	$D \leftarrow C - D$	$TX = \text{MUX}(T2, B : k_t)$
13	$XX \leftarrow D \times B$			$XX = \text{MUX}(X2, A : k_t)$
14	$YX \leftarrow A \times C$			$YX = \text{MUX}(Y2, D : k_t)$
15	$ZX \leftarrow D \times C$			$ZX = \text{MUX}(Z2, C : k_t)$
Post Computation				
	$z_2 \leftarrow z_2^{p-2}$			
	$x_2 \leftarrow x_2 \times z_2$			
	$y_2 \leftarrow y_2 \times z_2$			

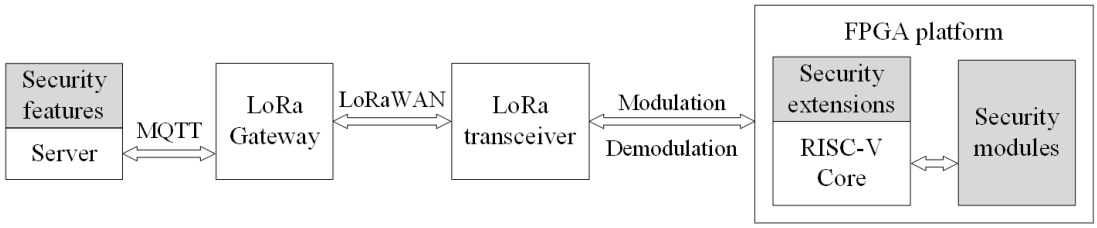


Fig. 4. Security cryptographic processor architecture and LoRaWAN Communication system.

it feasible to deploy elliptic-curve cryptography and maintain high performance, FPGA has low cost and power consumption features compared with high-end CPU and GPU, making it suitable to optimize the performance of IoT [34]. Therefore, a systematic solution for LoRaWAN secure communication is proposed with the FPGA platform, as shown in Figure 4.

In Figure 4, there are multi-layered solutions, including protocol, architecture, and cryptology. The standard LoRaWAN protocol lacks public-key support, and the proposed solution adds public-key cryptography to enhance security. The server enables connectivity, device monitoring, and end-user applications. The LoRa gateway transmits data between end devices and the cloud server. The gateway provides internet access and uses the MQTT protocol to send the packets. The transceiver offers long-range communication and communicates with the LoRa gateway. The proposed FPGA platform has a RISC-V core for software programming. Security extensions are inherent as the root of trust. Security modules are designed for public-key cryptography and security computations. These modules are packed as a coprocessor. The FPGA platform provides communication interfaces to interact with the LoRa transceiver directly. On the server side, ECC algorithms are also added to interact with the LoRa node to complete the key exchange and digital signature verification.

IV. HARDWARE ARCHITECTURE OF SECURE PROCESSOR

A. Overview of secure processor architecture

The secure processor architecture is presented in Figure 5. The architecture contains a RISC-V core, on-chip memory, peripherals, interconnections, and extended security modules. The processor is developed based on the Pulpino SoC platform. The developed architecture is to use the zero-riscy core

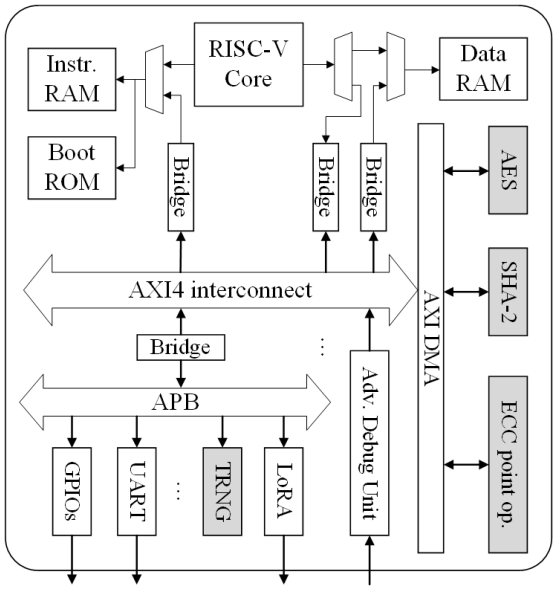


Fig. 5. Secure processor architecture.

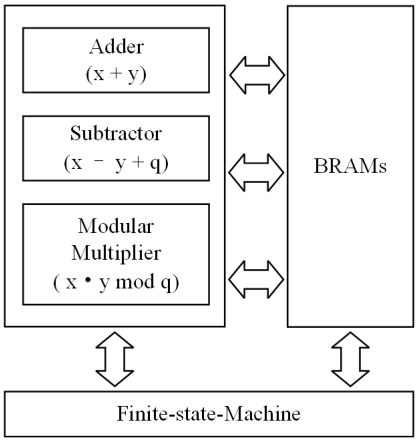


Fig. 6. Hardware architecture of ECC point operation.

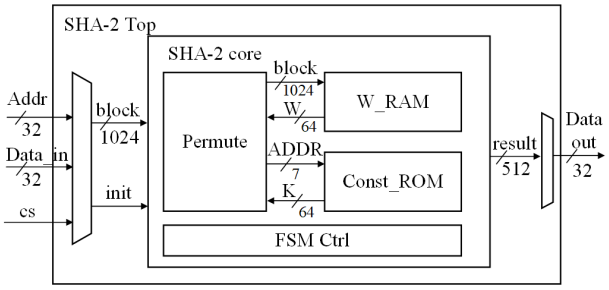


Fig. 7. Hardware architecture of SHA-2.

with standard 32 GPRs. The platform boots from the boot ROM. The executable code is pre-loaded into the RAMs and executed by the RISC-V core. The RISC-V core fetches instructions from the instruction RAM and reads/writes the data from/to the data RAM. The FPGA device contains a relatively larger number of block memory and some distributed memory implemented with LUT. Hence, we use the Block RAMs (BRAM) to function as the instruction RAM and data RAM.

The AIX bus is the main interconnection of different modules. The relatively low-speed peripherals (e.g., UART, GPIOs) are connected to the APB with a bridge to the AXI. The high-performance security modules, including AES, SHA-2, and ECC point operation modules, are connected with the AXI bus via the AXI DMA. The DMA provides high-bandwidth direct memory access between the RISC-V processor and security modules, so the data transmission will not become the bottleneck. The LoRa transceiver receives the settings via an SPI interface as a slave device. The SPI interface has a relatively low data bits rate, so a hardware SPI master is designed to parse the settings from the APB bus into an SPI protocol. The AES module is to accelerate the original LoRa node encryption, while the SHA-2 and ECC point operation modules are to support the public-key computations.

The True Random Number Generator (TRNG) module generates the root of trust for the secure processor. The TRNG is used in several cases: the ECDH key exchange algorithm uses a 256-bit random number as the private key, and the AES algorithm uses the exchanged key for encryption; the EdDSA signature algorithm requires a 256-bit random number to generate key pairs. Hence, the TRNG module safeguards the security of data and LoRaWAN applications and helps to build trust in the overall platform. The TRNG is implemented with a purely digital circuit and embedded inside the processor to ensure inaccessibility. The TRNG applies the staged-running Self-timed Ring (STR) architecture, which generates high-quality unpredictable random numbers by utilizing the diverse timing response to different initial values. The details of the TRNG architecture are described in [35].

B. Hardware architecture of security modules

The specific instance of the ECDH scheme is the X25519, and the EdDSA scheme is the Ed25519. The X22519 uses the x coordinate of the Curve25519, and the Ed25519 uses a curve birationally equivalent to Curve25519. Scalar multiplication is the most time-consuming function and a unified architecture is designed for both schemes, as shown in Figure 6. The inside modular multiplication architecture is designed according to Algorithm 1. The timing schedule of X25519 and Ed25519 are shown in Table I and Table II. The designed hardware module is lightweight, and the arithmetic unit contains an Adder, a Subtractor, and a Modular Multiplier. The inside computation bit width is 32-bit. The BRAMs are used to store intermediate computation results. A finite state machine (FSM) is used to control the calculation flow, and the calculation is designed to run in constant time to avoid simple timing attacks.

The SHA-2 module supports both SHA_{256} and SHA_{512}. For the SHA_{256}, the input block is 512-bit, the inside permutation is performed with 32-bit, and the output result is 256-bit. For the SHA_{512}, the input block is 1024-bit, the permutation bit is 64-bit, and the output block is 512-bit. In order to be compatible with SHA_{512}, for SHA_{256}, the higher significant bytes are discarded. A distributed ROM is used to store the constants for the round permutation. A distributed RAM

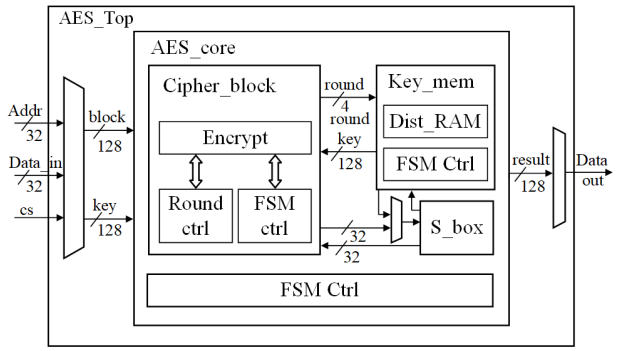


Fig. 8. Hardware architecture of AES encryption.

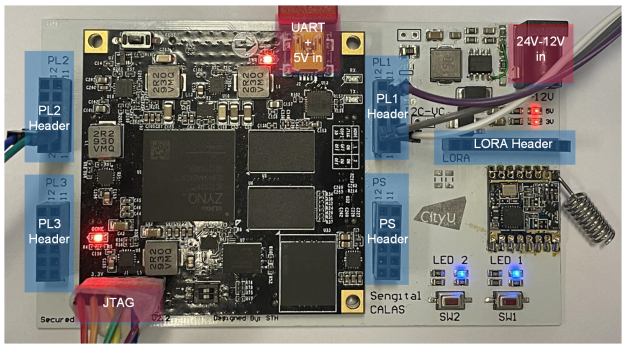


Fig. 9. The prototype of the development board.

is used to store the intermediate computation results. The computation flow is controlled using an FSM.

The AES module supports the AES-128 operation used in the LoRaWAN encryption standard. The AES core contains a cipher block and a key management block. To begin with, the 128-bit shared key is sent and stored in distributed RAM. Next, the 128-bit input is sent for each block encryption. The S-box is used to expand the key for encryption.

V. Evaluations and Discussions

A. Development board prototype and experiment setup

A customized tiny FPGA platform is developed for further validation experiments on IoT applications, as shown in Figure 9. This platform includes an xc7z020clg FPGA chip and a LoRa transceiver chip SX1262. The FPGA bitstream is downloaded onto the board over JTAG. The USB-UART device is attached to the corresponding ports in the FPGA board and performs online programming (TX) or prints out the information (RX). The hardware is designed with Verilog HDL, and the hardware bitstream is generated in Xilinx Vivado 2019.2. The resource consumption result is shown in Table III. The hardware resources of three security modules are tested before integration into the processor, and the secure processor includes all modules used in the application. The application software is designed using the C/C++ language and programmed into the memory on the board through the UART port. The prototype board is configured to run at 50 MHz. The power consumption results are accurately simulated from the Vivado. The static power is 0.116 W, and the dynamic power is 0.206 W. The total on-chip power is 0.321 W.

TABLE III
Hardware resources consumption in FPGA

HW Module	LUT	Slice	FF	BRAM	DSP
AES	1039	340	307	4.5	0
SHA-2	3164	857	2064	0	0
ECC point op.	2153	621	611	8	4
Secure processor[a]	24414 (45%)	9883 (74%)	26837 (25%)	113 (80%)	5 (2%)

[a]The secure processor contains all the modules in Figure 5, including the RISC-V core, interconnection, memory, and extended security modules.

B. Speed performance and memory efficiency

The firmware program is designed to configure the hardware module, including the TRNG, AES, SHA-2, and ECC point operation modules. To test the performance and efficiency of the designed platform, the results of SW design and SW/HW co-design are compared, as shown in Table IV. The SW design is to run the whole function in the RISC-V core. The SW/HW co-design also runs the function in the RISC-V core, but at the same time, the firmware program is used to call the hardware modules to accelerate the functions.

The tested AES encrypt function encrypts one block (128-bit) message with the (ECDH) exchanged key. The AES achieves the most basic security functions and can be used before and after the security extension of the LoRa network. The AES is more lightweight compared with other public-key functions. Our SW/HW co-design can achieve 12.1x memory efficiency and 8.1x speed-up in the AES.

The tested ECDH function is the scalar multiplication function used in X255519. Since both public and shared key generation involve only one scalar multiplication function, this function is the only function that needs to be tested for ECDH. As shown in Table IV, both SW design and SW/HW co-design consume the same size of data RAM. However, the instruction RAM size is significantly larger in the pure SW design due to the high complexity of scalar multiplication. Our SW/HW co-design can achieve 12.3x memory efficiency and 144.7x speed-up in the ECDH.

The EdDSA is the most complex function in our security extension. The SHA_{512} and scalar multiplication functions are the most time-consuming. The tested SHA_{512} function is to hash the 32 bytes private key and 32 bytes message, which consumes 1.08 ms in pure software design. Our SW/HW co-design can achieve 4.8x memory efficiency and 5.6x speed-up in the SHA_{512}. The scalar multiplication is slightly more complex in EdDSA compared with ECDH since the two coordinates scalar multiplication is involved. The tested software design of the EdDSA scalar multiplication performs a lot of precomputing and stores the results in the data RAM to speed up the function. However, its speed improvement is limited and consumes many RAM resources. Our SW/HW co-design can achieve 93.0x memory efficiency and 23.8x speed-up in the EdDSA scalar multiplication function. For the other functions, including the modular reduction, number addition, and concatenation, they are kept to run in software

TABLE IV
EXPERIMENT RESULTS ON SPEED AND MEMORY SIZE

Function	Pure SW design			SW/HW co-design		
	Inst RAM (kB)	Data RAM (kB)	Time (ms)	Inst RAM (kB)	Data RAM (kB)	Time (ms)
AES_encrypt	22.6	2.6	0.57	2.0	0.066	0.07
SHA_{512}	29.4	2.1	1.08	6.2	0.294	0.19
ECDH	117.0	3.2	645.8	6.5	3.2	4.46
EdDSA_scalar_mul	174.6	77.0	168.7	2.0	0.704	7.07
EdDSA_sign	422.6	81.6	171.8	208.2	0.864	8.53

TABLE V
COMPARISON WITH RELATED WORKS

Design	Algorithm	Device	LUT/Slice/DSP	10^3 Cycles	LUT/Slice/DSP $\times 10^6$ Cycles	Static/Dynamic Power (mW)
[36]- Single Core	X25519	XC7Z020	2783/ 1029/ 20	79.4	220/ 81/ 1.588	N/A
[36]- Multi Core			34009/ 11277/ 220	34.0	1156/ 383/ 7.480	
[37]	X25519	Zynq7030	26483/ 8639/ 260	13.6	360/ 117/ 3.536	150 / 789
[38]	X25519	Zynq 7020	2707/ 775/ 15	75	203/ 58/ 1.125	N/A
	Ed25519		11148/ 3204/ 16	132	1471/ 442/ 2.112	
This work	X25519	Zynq7000 XC7Z020	2153/ 621/ 4	223	480/ 138/ 0.892	116 / 206
	Ed25519		5317/ 1478/ 4	426	2265/ 629/ 1.704	

in the overall SW/HW co-design. Our SW/HW co-design can achieve 2.4x memory efficiency and 20.1x speed-up for the complete EdDSA function. The consumed instruction RAM size of the EdDSA function is larger than other functions, but our current memory budget is still enough to support this function. If a more tight memory budget is required in the future, the modular reduction and number addition can be implemented in FPGA to reduce the memory size further.

C. Results comparison and discussion

In terms of security enhancement, the classical LoRaWAN system calculates the AES key with a pre-installed root key. Our enhanced system introduces ECDH for key exchange before encryption. The pure software method takes 0.57 ms for each 128-bit AES encryption. With the hardware acceleration, the LoRa node spends 4.46 ms for key exchange and performs multiple AES encryption afterward. For the messages which require data authentication, the LoRa node takes 8.53 ms to generate a digital signature in each LoRa frame. The signature is then sent to the server for verification. Since the classical LoRaWAN system lacks public-key based key exchange and digital signatures, our system provides better security guarantees.

Some works also tried to enhance the security of LoRa communication. Tomasin et al. [39] identified the possibility of regenerating the device nonce. The author proposed a random number generator algorithm and increased the devise nonce size. Kim and Song [40] proposed using a second root key to avoid the application and network session keys generated from the same root key. In comparison, our platform generates the random number more securely with a digital TRNG core, which has passed the NIST test with high p-values, as described in [35]. With public-key cryptography algorithms, we can safely update the root key through insecure channels and provide message authentication. Hence, our solution is better in terms of security.

The hardware design results are also compared with related works, as shown in Table V. Sasdrich et al. [36] proposed high-performance single-core and multi-core architecture for X25519. Our lightweight design consumes fewer resources but runs at a lower speed. Our design achieves a better area-time product than the multi-core architecture but slightly worse than the single-core architecture. Kopperman et al. [37] proposed a low-latency design of X25519, which consumed more than 10x the resources compared with our work. Both the design in [37] and our work are in Xilinx Zynq architecture. In contrast, our design removes the ARM processor and implements the RISC-V core in FPGA. Compared with the work in [37], our design consumes only 77% of the static power and 26% of the dynamic power. Turan et al. [38] combined Ed25519 and X25519 in a single module, which is similar to our design. In comparison, our design for lightweight implementation consumes half the resources but 3x as many cycles. Overall, our design is the most compact compared with previous work and achieves a relatively similar area-time product. Our design consumes lower power compared to similar SoCs. Therefore, our design is suitable for resource-constrained, multi-functional, and low-power IoT systems.

VI. CONCLUSIONS

In this paper, we propose a security-enhanced LoRaWAN communication network with public-key infrastructure. We enhance the key exchange process with the ECDH algorithm and data authentication with the EdDSA signature algorithm. The extended protocol is compatible with the LoRaWAN. We propose a RISC-V based secure processor architecture and provide a systematic solution for the secure network. We provide the root of trust with a digital TRNG core

and design hardware modules to accelerate the computation-intensive security algorithms. We prototype and evaluate a development board on a practical LoRa communication system. The measured total on-chip power is 0.321W. The proposed architecture can achieve a 5.6x-144.7x speed up and reduce memory usage by 2.4x-12.3x.

Our future work is to develop an application-specific integrated circuit (ASIC) and replace the existing processor in the application with the designed secure processor for secure communication.

VII. Acknowledgements

The author would like to thank Mr. Jack Junjie Liu, Mr. John Yiqing Zhang, Mr. Man-Kit Sit, Mr. Max Tsz-Ho Sze, and Ms. Yifei Zhao for their great help in this work. The author would also like to thank the anonymous reviewers for their valuable comments.

References

[1] A. Shrivastava, K. Murali Krishna, M. Lal Rinawa, M. Soni, G. Ramkumar, and S. Jaiswal, "Inclusion of iot, ml, and blockchain technologies in next generation industry 4.0 environment," *Materials Today: Proceedings*, vol. 80, pp. 3471–3475, 2023.

[2] S. Balaji, K. Nathani, and R. Santhakumar, "Iot technology, applications and challenges: A contemporary survey," *Wireless personal communications*, vol. 108, no. 1, pp. 363–388, 2019.

[3] A.-A. A. Boulogeorgos, P. D. Diamantoulakis, and G. K. Karagiannidis, "Low power wide area networks (lpwans) for internet of things (iot) applications: Research challenges and future trends," *arXiv preprint arXiv:1611.07449*, 2016.

[4] G. S. Ramachandran, F. Yang, P. Lawrence, S. Michiels, W. Joosen, and D. Hughes, "μpnp-wan: Experiences with lora and its deployment in dr congo," in *2017 9th International Conference on Communication Systems and Networks (COMSNETS)*, 2017, pp. 63–70.

[5] I. d. S. Batalha, A. V. R. Lopes, W. G. Lima, *et al.*, "Large-scale modeling and analysis of uplink and downlink channels for lora technology in suburban environments," *IEEE Internet of Things Journal*, vol. 9, no. 23, pp. 24 477–24 491, 2022.

[6] S. R. J. Ramson, W. D. León-Salas, Z. Brecheisen, *et al.*, "A self-powered, real-time, lorawan iot-based soil health monitoring system," *IEEE Internet of Things Journal*, vol. 8, no. 11, pp. 9278–9293, 2021.

[7] W. Xu, J. Y. Kim, W. Huang, S. S. Kanhere, S. K. Jha, and W. Hu, "Measurement, characterization, and modeling of lora technology in multifloor buildings," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 298–310, 2020.

[8] S. Devalal and A. Karthikeyan, "Lora technology - an overview," in *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, 2018, pp. 284–290.

[9] N. Sornin, M. Luis, T. Eirich, T. Kramp, and O. Hersent, "Lorawan specification," *LoRa alliance*, 2015.

[10] S. Naoui, M. E. Elhdhili, and L. A. Saidane, "Enhancing the security of the iot lorawan architecture," in *2016 International Conference on Performance Evaluation and Modeling in Wired and Wireless Networks (PEMWN)*, 2016, pp. 1–7.

[11] J. Han and J. Wang, "An enhanced key management scheme for lorawan," *Cryptography*, vol. 2, no. 4, 2018, ISSN: 2410-387X.

[12] W. Xu, S. Jha, and W. Hu, "Lora-key: Secure key generation system for lora-based network," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6404–6416, 2018.

[13] A. K. Junejo, F. Benkhelifa, B. Wong, and J. A. Mccann, "Lora-lisk: A lightweight shared secret key generation scheme for lora networks," *IEEE Internet of Things Journal*, vol. 9, no. 6, pp. 4110–4124, 2022.

[14] N. Torres, P. Pinto, and S. I. Lopes, "Security vulnerabilities in lpwans—an attack vector analysis for the iot ecosystem," *Applied Sciences*, vol. 11, no. 7, 2021.

[15] Z. Sun, H. Yang, K. Liu, Z. Yin, Z. Li, and W. Xu, "Recent advances in lora: A comprehensive survey," *ACM Trans. Sen. Netw.*, vol. 18, no. 4, Nov. 2022.

[16] J. Jung, B. Kim, J. Cho, and B. Lee, "A secure platform model based on arm platform security architecture for iot devices," *IEEE Internet of Things Journal*, vol. 9, no. 7, pp. 5548–5560, 2022.

[17] H. B. Amor, C. Bernier, and Z. Přikryl, "A risc-v isa extension for ultra-low power iot wireless signal processing," *IEEE Transactions on Computers*, vol. 71, no. 4, pp. 766–778, 2022.

[18] F. Taheri, S. Bayat-Sarmadi, and S. Hadayeghparast, "Risc-hd: Lightweight risc-v processor for efficient hyperdimensional computing inference," *IEEE Internet of Things Journal*, vol. 9, no. 23, pp. 24 030–24 037, 2022.

[19] K. Asanović and D. A. Patterson, "Instruction sets should be free: The case for risc-v," *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2014-146*, 2014.

[20] M. Gautschi, P. D. Schiavone, A. Traber, *et al.*, "Near-threshold risc-v core with dsp extensions for scalable iot endpoint devices," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 10, pp. 2700–2713, 2017.

[21] *Pulpino*, 2019. [Online]. Available: https://github.com/pulp-platform/pulpino.

[22] M. Eldefrawy, I. Butun, N. Pereira, and M. Gidlund, "Formal security analysis of lorawan," *Computer Networks*, vol. 148, pp. 328–339, 2019.

[23] E. Aras, G. S. Ramachandran, P. Lawrence, and D. Hughes, "Exploring the security vulnerabilities of lora," in *2017 3rd IEEE International Conference on Cybernetics (CYBCONF)*, 2017, pp. 1–6.

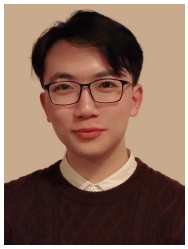
[24] J. P. Shanmuga Sundaram, W. Du, and Z. Zhao, "A survey on lora networking: Research problems, current solutions, and open issues," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 371–388, 2020.

[25] D. Basu, T. Gu, and P. Mohapatra, "Security issues of low power wide area networks in the context of lora networks," *ArXiv*, vol. abs/2006.16554, 2020.
[26] 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, pp. 129–140.
[27] Arjan, *Fair use policy explained*, May 2021. [Online]. Available: https://www.thethingsnetwork.org/forum/t/fair-use-policy-explained/1300.
[28] Z. Vahdati, S. Yasin, A. Ghasempour, and M. Salehi, "Comparison of ecc and rsa algorithms in iot devices," *Journal of Theoretical and Applied Information Technology*, vol. 97, no. 16, 2019.
[29] S. Bai, L. Ducas, E. Kiltz, *et al.*, "Crystals-dilithium–algorithm specifications and supporting documentation," *NIST Post-Quantum Cryptography Standardization Round 3*, 2020.
[30] D. J. Bernstein, "Curve25519: New diffie-hellman speed records," in *Public Key Cryptography - PKC 2006*, M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 207–228.
[31] L. Chen, D. Moody, A. Regenscheid, and K. Randall, "Recommendations for discrete logarithm-based cryptography: Elliptic curve domain parameters," National Institute of Standards and Technology, Tech. Rep., 2019.
[32] A. Langley, M. Hamburg, and S. Turner, "Rfc 7748: Elliptic curves for security," *Internet Research Task Force (IRTF)*, 2016.
[33] S. Josefsson and I. Liusvaara, *Rfc 8032: Edwards-curve digital signature algorithm (eddsa)*, 2017.
[34] R. Nair, P. Sharma, and T. Sharma, "Optimizing the performance of iot using fpga as compared to gpu," *Int. J. Grid High Perform. Comput.*, vol. 14, no. 1, pp. 1–15, Jun. 2022.
[35] Y. Liu, R. C. C. Cheung, and H. Wong, "A bias-bounded digital true random number generator architecture," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, pp. 133–144, 2017.
[36] P. Sasdrich and T. Güneysu, "Efficient elliptic-curve cryptography using curve25519 on reconfigurable devices," in *Reconfigurable Computing: Architectures, Tools, and Applications*, D. Goehringer, M. D. Santambrogio, J. M. P. Cardoso, and K. Bertels, Eds., Cham: Springer International Publishing, 2014, pp. 25–36.
[37] P. Koppermann, F. De Santis, J. Heyszl, and G. Sigl, "X25519 hardware implementation for low-latency applications," in *2016 Euromicro Conference on Digital System Design (DSD)*, 2016, pp. 99–106.
[38] F. Turan and I. Verbauwhede, "Compact and flexible fpga implementation of ed25519 and x25519," *ACM Trans. Embed. Comput. Syst.*, vol. 18, no. 3, Apr. 2019.
[39] S. Tomasin, S. Zulian, and L. Vangelista, "Security analysis of lorawan join procedure for internet of things networks," *2017 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pp. 1–6, 2017.
[40] J. Kim and J. Song, "A dual key-based activation scheme for secure lorawan," *Wireless Communications and Mobile Computing*, 2017.

Gaoyu Mao received the B.Eng. degree in integrated circuit design and integration system from School of Microelectronics, Shandong University, jinan, China in 2020. He is currently working toward the Ph.D. degree in the Department of Electrical Engineering, City University of Hong Kong. He visited the Zhejiang Lab in Hang Zhou from September to November of 2022. His research interests include reconfigurable computing with FPGA, and cryptographic hardware design.

Yao Liu (Member, IEEE) received the B.S. and M.S. degrees in microelectronics from Fudan University, Shanghai, China, in 2008 and 2011, respectively, and the Ph.D. degree from the Department of Electrical Engineering, City University of Hong Kong (CityU), Hong Kong, in 2019. Before he obtained a faculty job, he gained over-5-year working experience in IC industry and academia. He is currently an Assistant Professor with the School of Microelectronics Science and Technology, Sun Yat-Sen University. His research interests include FPGA prototyping and VLSI implementation for hardware security systems, computer architecture, and network systems.

Wangchen Dai received the B.Eng. degree in electrical engineering and automation from Beijing Institute of Technology, China, in 2010, the M.A.Sc. degree in electrical and computer engineering from the University of Windsor, Canada, in 2013, and the Ph.D. degree in electronic engineering from the City University of Hong Kong in 2018. After completing the Ph.D. study, he had appointments at Hardware Security Lab, Huawei Technologies Company Ltd., in 2018, and the Department of CSSE, Shenzhen University in 2020, respectively. He is currently working as a Senior Researcher with Zhejiang Lab, Hangzhou, China. His research interests include cryptographic hardware and embedded systems, fully homomorphic encryption, and reconfigurable computing.



Guangyan Li received the B.Eng degree in 2020 from the Department of Electrical Engineering, City University of Hong Kong. He joined the overseas internship scheme to the LIRMM at Montpellier, France from June to August of 2019. He was a research-based FYP student under the supervision of Dr. Ray C. C. Cheung. He started his Ph.D. study in CityU EE department in 2020/21. His research interests include reconfigurable computing with FPGA, and hardware security.

Zhewen Zhang received the B.Eng degree in electronics information science and technology from School of Astronautics, Harbin Institute of Technology, Harbin, China in 2020. She is currently working towards the Ph.D. degree in the Department of Electrical Engineering, City University of Hong Kong. Her research interests include processor design, and hardware security.

Alan H. F. Lam (Member, IEEE) is an Adjunct Professor with the Department of Electrical Engineering, College of Engineering, City University of Hong Kong, and an Adjunct Professor with the Department of Mechanical and Automation Engineering, Faculty of Engineering, The Chinese University of Hong Kong. His main research interest is on artificial intelligence, the Internet of Things, and the use of wearable motion sensors for sports science and medical applications. He has won numerous prestigious local and international awards for product innovation and excellence. He was named one of the five local innovation heroes by the Hong Kong Science Park in 2014 for his excellence in Research and Development projects commercialization and was selected as one of the awardees of 2015 Hong Kong Ten Outstanding Young Persons.

Ray C.C. Cheung (Senior Member, IEEE) received the B.Eng. (Hons.) and M.Phil. degrees in computer engineering and computer science and engineering from The Chinese University of Hong Kong, and the DIC and Ph.D. degrees in computing from Imperial College London. He received the Hong Kong Croucher Foundation Fellowship for his Postdoc research work at the Electrical Engineering Department at UCLA, and completed his visiting fellowship at Princeton University. He is a Professor at the Department of Electrical Engineering, and an Associate Provost (Digital Learning) at City University of Hong Kong. His current research interests include cryptographic processor designs and embedded system designs. He served as the Technical Chair of FPT'02, General Chair of ARC'12, and General Co-Chair of FPT'22. He is currently the Treasurer of the IEEE Hong Kong Section.