Hindawi
Security and Communication Networks
Volume 2022, Article ID 9709601, 17 pages
https://doi.org/10.1155/2022/9709601

Research Article

An Efficient Lightweight Cryptographic Instructions Set Extension for IoT Device Security

Wajih El Hadj Youssef, Ali Abdelli, Fethi Dridi, Rim Brahim, and Mohsen Machhout

Faculty of Sciences of Monastir, Electronics and Micro-Electronic Laboratory (LEME), Monastir 5000, Tunisia

Correspondence should be addressed to Wajih El Hadj Youssef; elhadjyoussef.wajih@gmail.com

Received 5 June 2021; Revised 6 December 2021; Accepted 4 January 2022; Published 5 February 2022

Academic Editor: Ricardo Chaves

The Internet of Things is changing all sectors such as manufacturing, agriculture, city infrastructure, and the automotive industry. All these applications ask for secure processors that can be embedded in the IoT devices. Furthermore, these devices are restricted in terms of computing capabilities, memory, and power consumption. A major challenge is how to meet the need for security in such resource-constrained devices. This paper presents a customized version of LEON3, the ReonV RISCV (Reduced Instruction Set Computer-five) processor, dedicated for IoT applications that has strong effective security mechanisms built in at the design stage. Firstly, efficient lightweight cipher designs are elaborated and validated. Then, the proposed cryptographic instructions (PRESENT and PRINCE) are integrated into the default instruction set architecture of the ReonV processor core. The instruction set extensions (ISE) of lightweight cipher modules can be instantiated in software routines exactly as the instructions of the base architecture. A single instruction is needed to implement a full lightweight cryptographic instruction. The customized ReonV RISCV processor is implemented on a Xilinx FPGA platform and is evaluated for Slice LUTs plus FF-pairs, frequency, and throughput. Obtained results show that our proposed concepts not only can achieve good encryption results with high performance and reduced cost but also are secure enough to resist against the most common attacks.

1. Introduction

The Internet of Things (IoT) refers to a huge number of connected devices to the Internet, all collecting and sharing vast amounts of data [1]. Regarding the huge amount of connected devices, the IoT has become an integral part of the lives of billions of people around the world.

However, when using open and connected devices such as smart city assets like smart transportation, smart traffic-lights, and smart meters, in industry 4.0 devices like programmable logic controllers (PLCs), robots, and machines, there have been recently many IoT-related security threats with a devastating security incident and the problems will get worse [2, 3]. Hence, we all have to make security the key issue in the choice and deployment of IoT-related devices.

Cryptography is an effective countermeasure, and the IoT is now required to apply encryption to autonomous devices in environments with various restrictions. Lightweight cryptography is a technology researched and developed to enable the application of secure encryption, even for devices with limited resources [4].

Securing IoT devices require innovation at the processor level to boost the next generation of IoT edge devices with a new kind of enhanced processors. This category of processors is built on new levels of Central Processing Unit (CPU) with integrated IoT features, real-time performance, and functional safety. Security mechanism can be added to processor based IoT devices by means of specific software libraries or dedicated hardware accelerators. The software approach is very flexible, but it is not suited for extremely constrained devices. Indeed, software solution can be too costly in terms of memory occupation or energy consumption and the performance may be enough to meet the application requirements. Moreover, certain security threats may not be defeated using software solutions only. The second approach allows meeting better performance and

often better security but is not flexible and often requires an excessively large area occupation. Furthermore, depending on the frequency and the amount of data which needs to be moved to the accelerator, the delay of the transfer could overshadow the speedup of the dedicated hardware.

This paper focuses on the addition of custom lightweight cryptographic instructions into the default instruction set architecture of the ReonV processor.

This work has led to the creation of ReonV-LWCIS processor, which is a new version of the ReonV. It represents a modified version of the LEON3. The ReonV is a synthesizable VHDL model of a 32 bit processor originally compliant with the SPARC V8 architecture, changed to implement the RISC-V RV32I ISA. Two instructions of lightweight cryptographic algorithms: PRESENT and PRINCE, are incorporated in the customized processor with respect of computing capabilities, cost, efficiency (i.e., throughput per slice) power, energy per bit consumption, and security level.

The rest of this article is organized as follows: the next section discusses earlier works related to instruction set extensions for cryptographic algorithms. Section 3 gives a description of LEON3 SPARC V8 processor and the modified version of LEON3, the ReonV RISCV processor. The elaborated hardware architectures of lightweight cryptographic algorithms are analyzed in Section 4. The proposed instruction set extensions for lightweight cryptographic instructions (LWCIS) are analyzed in Section 5. Synthesis results on FPGA (Field Programmable Gate Arrays) platforms as well as security analysis of the elaborated cipher designs are achieved in Section 6. Implementation results of the customized processor are presented in this section too. Section 7 concludes this paper.

2. Related Works

This section reviews works related with our proposed approach. Numerous implementation approaches often exist for a given lightweight cryptographic algorithm. The software approach consists of using general purpose processors together with embedded software implementations. The second approach is the implementation using dedicated hardware platforms such as FPGA or ASIC (Application Specific Integrated Circuit). Another hybrid approach is based on instruction set extensions (ISEs) technique and consists of the customization of the processor's instruction set and the microarchitecture.

The work presented in [5] introduced a software implementation of HIGHT lightweight block cipher, on 8-bit AVR and 32-bit ARM Cortex-M3 and hardware (i.e., ASIC). The authors suggested generic methods which can be easily used for other ARX block ciphers.

In [6], Varici et al. have implemented different design methodologies of PRESENT algorithm which include hand-coded RTL, Vivado HLS, PicoBlaze, VerySimpleCPU (VSCPU) based microcontrollers, and a customized VSCPU. They propose a customized CPU design based on optimizing the instruction set architecture for PRESENT algorithm. They prove that their solution is efficient with better flexibility features compared to RTL designs.

Eisenkraemer et al. [7] have focused on the enhancement of the instruction set architecture (ISA). A hybrid design is applied to boost the performance of their design. The recommended design is validated by measuring the area overhead, memory parameters, and the speedup for optimized implementations of AES, DES, 3DES, and SHA using Cadence LX7 Processor and Xtensa platform. Their obtained results provide excellent tradeoff with the area, memory, and cycle count performances.

In another work [8], authors have investigated a separate instruction set extension for 32- and 64-bit base architectures. Their results show better performances for an AES-128 block encryption compared to software implementation. The improvements are about 4x and 10x faster with a hardware cost of 1.1 K and 8.2 K gates, respectively. Their work has supported RISC-V in becoming the first widely implemented ISA to support AES acceleration across all implementation profiles, from embedded IoT devices to application and server class processors.

In [9] Werner et al. have presented a method for software IP confidentiality and authentic execution on IoT devices. Their method named Sponge-based Control Flow Protection (SCFP) consists of hardware extension between the CPU's fetch and decode stage to decrypt and authenticate instructions at the compile time. This prevents code-reuse, code-injection, and fault attacks on the code and the control flow. To check their concept, they proposed an extended version of RISC-V processor enhanced with Sponge-based Control Flow Protection. They achieved an average overhead of SCFP in code size of 19.8% and an execution time of 9.1%. They claimed that their design meets IoT devices requirements.

The works of Grabher et al. [10] described a number of lightweight instruction set extensions (ISEs). They have presented a design that can improve the performance overhead that results from bit-sliced implementation, at the same time preserving the advantages of bit-sliced implementation. They have suggested a generic design with a high agility degree of cryptographic algorithms (AES, DES, Serpent, and PRESENT), SHA-1 hash function, and polynomials multiplication.

In [11] a processor instruction set based on stack-based processor architecture is presented. The recommended design consists of connecting a serialized Klein lightweight cryptographic cipher coprocessor to the 32-bit ZPU core. By this way, different lightweight security and authentication primitives can be implemented in a code and area effective way. The proposed design is synthesized using VeriSilicon GSMC 0.13 um with a frequency of 100 kHz and a gate count of 4.5 K GE.

In another work, an ongoing effort in securing RISC-V core called S-RISC-V is introduced [12]. The proposed methodology consists of using key generation to protect memory with ISA extension and hardware implementation. The suggested architecture has been verified on Zedboard FPGA platform, driven by the host ARM core with a frequency of at 25 MHz. Compared to the original RISC-V core, they noticed an area overhead less than 10%.

Steinegger and Primas [13] proposed an instruction set extension for Ascon-p. Their proposed design consists of integration into a 32-bit RI5CY processor core. This permits accelerating the symmetric cryptographic computations with a low cost. The proposed instruction extension for RISC-V avoids the need for protection against implementation attacks, by the choice of the appropriate AEAD mode in software. They prove that their proposed accelerator can be achieved with hardware metrics of 4.7 kGE. Park et al. in [14] have investigated a set of implementation methods of the Simeck algorithms with different data block sizes using a 64-bit Intel processor Advanced Vector Extension 2 (AVX2). They proposed an efficient encryption method for human care service. They obtained 3.5 cycles/byte and 4.6 cycles/byte for Simeck32/64 and Simeck64/128 encryption, respectively.

In industry, lightweight cryptographic module is used by NXP semiconductors. The security system on LPC55S6x devices is ensured by PRINCE encryption algorithm. The cipher is used for real-time encryption of data being written to the flash chip and decryption of secured on-chip flash data while reading [15]. They proposed a real-time on-the-fly encryption/decryption functionality executed in one cycle. This permits securing application code, as well as stored keys and flash update security.

Because innovative IoT devices often times require customizable security solutions, there is a room to elaborate custom cryptographic architectures with better implementations performances. Processors customization is still an active area of research which aims at reducing the gap between the application requirements and the features of standard processor by extending the base instruction set of embedded processors with a number of instructions dedicated to security.

In this work, we suggest a solution that enclose the efficiency of lightweight cryptographic algorithms, the software-based implementations flexibility, and the high performance of custom hardware architectures. The proposed solution consists in developing an enhanced processor architecture by customizing both the microarchitecture and the instruction set of the processor. Two lightweight cryptographic instructions set extensions (PRESENT and PRINCE) for RISC-V ReonV processor core are elaborated. These custom instructions can then be instantiated in software routines exactly as the instructions of the base architecture.

2.1. Description of the ReonV and LEON3 Processors. To be suited for lightweight and IoT applications, microprocessor core has to be optimized in terms of area and power consumption. Many works treat with architectures and freeware processor designs. The main difficulty consists in applying a compact instruction set architecture which is also code-efficient and has GCC support. LEON3 [16] is a configurable processor written in VHDL, provided as part of the GRLIB IP Library licensed by Cobham Gaisler AB. The advantage of the availability of its source helps to make changes and explore new concepts. The synthesizable model is a 32-bit processor compliant with the SPARC V8 architecture and the full source code is available under the GNU GPL license, allowing free and unlimited use for research. LEON3 as a processor is not dedicated to the IoT; it works in space shuttles and domains that need heavy calculations.

In this work we used ReonV, a RV32I open-source CPU, which is a VHDL model that was developed by reusing the LEON3 processor as well as the structure of its SOC (System On Chip). It consists of changing only its 7-step integer instruction pipeline to implement the RV32I ISA instead of the original ISA SPARCV8. The instruction set architecture (ISA) of the original processor LEON3 is changed from SPARCV8 to RISC-V [18], while retaining all other IP cores and resources such as memory, memory controllers, peripheral support, debugging support unit (DSU), synthesis scripts for several FPGAs from different manufacturers, and others.

As shown in Figure 1, there is the SOC structure of the LEON3 processor in the GRLIB models. It also illustrates that there are many gains when reusing this structure in the development of a new RISC-V processor, because it will inherit without requiring changes.

The RISC-V architecture can be personalized for the IoT devices. It represents a flexible computing platform which enables novel development for IoT applications [19].

2.2. RISC-V VS SPARC V8. SPARC V8 [20] is suitable to be implemented on servers and not for embedded devices. It is under the GPL license. This architecture has limits; it lacks several important architectural features and other properties that dramatically increase the implementation complexity, especially for high-performance implementations.

Furthermore, contrary to SPARC V8, the RISC-V architecture is modular; a good balance between cost and efficiency can be achieved for a dedicated application. It is a free and open ISA that avoids technical errors and is simple to implement in many microarchitectural styles. Another reason why RISC-V has attracted industry interest is that it was designed to be both stable for long-term viability and adaptable to accommodate a lot of applications. A comparison of open ISAs is shown in Table 1.

2.3. Instruction Pipeline. ReonV and LEON3 both have a 7-stage pipeline. ReonV ended up using the same steps as LEON3 for convenience (IF, ID, RA, EX, ME, EXP, WB) [17]. For a 7-stage pipeline processor, the integer unit (UI) uses 7 cycles to complete an instruction:

(i) FI (fetch instruction): the CPU reads the instructions from the memory address whose value is present in the program counter.

(ii) DI (decode instruction): the instruction is decoded to identify the type of instruction and the register file is accessed to obtain the values of the registers used in the instruction.

(iii) RA (register access): this step allows the reading of the operands in the registers.

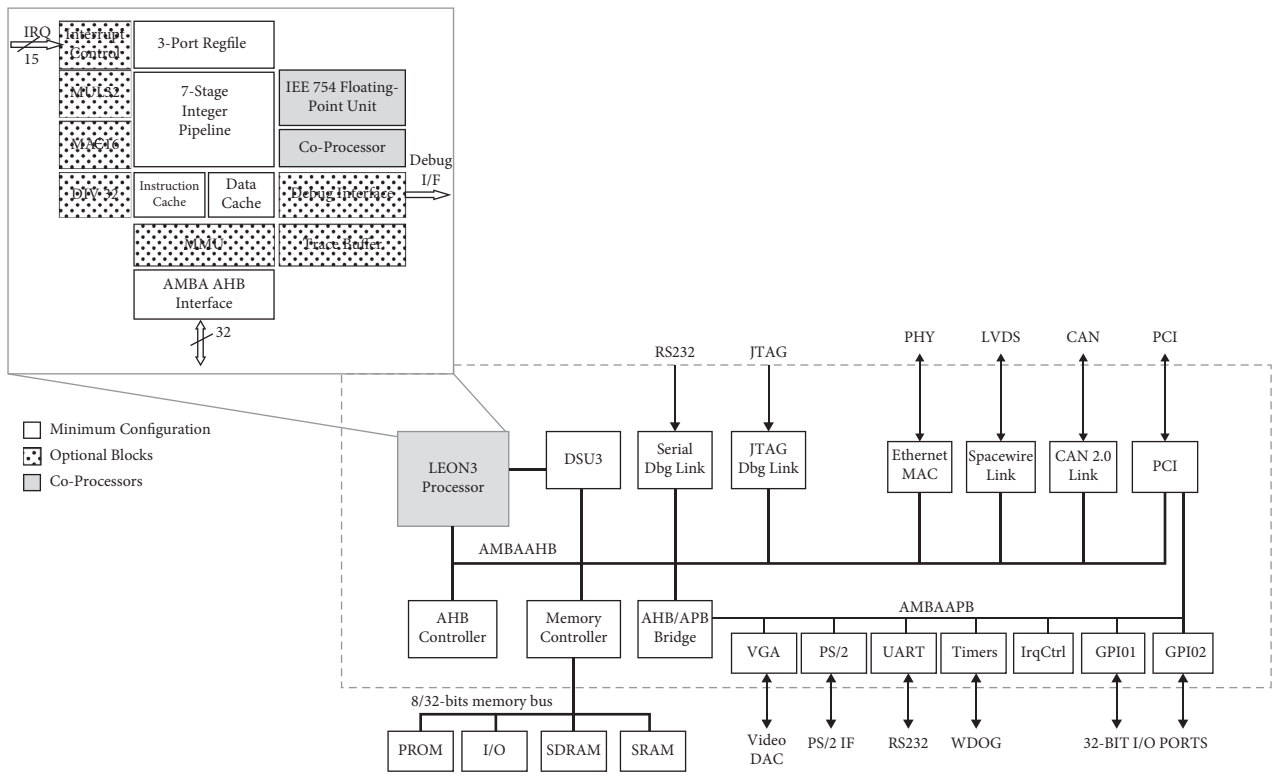

Figure 1: Representation of peripherals and SOC structure of LEON3.

Table 1: Differences between SPARC V8 ISA and RISC-V.

ISA	Base + ext	Compact code	Quad FP	32 bits	64 bits	128 bits	GCC
SPARC V8			x	x			x
RISC-V	x	x	x	x	x	x	X

(iv) EX (execution instruction): this step allows you to retrieve the operands from the registry file and execute the instruction using the ALU (Arithmetic-Logic Unit).

(v) ME (memory): the memory operands are read and written from and to the memory present in the instruction. This step denotes a transfer from a register to the memory in the case of a STORE type instruction (write access) and from memory to a register in the LOAD case (read access).

(vi) EXP (exception): this step propagates exceptions resulting from the execution of instructions.

(vii) WB (write back): the extracted value is rewritten in the register present in the instructions in order to benefit from a very short access time for the following instructions.

When designing integrated circuits for IoT applications, it is desirable to create optimized processors to efficiently run IoT algorithms having an effective resistant against external attacks with a tradeoff between high-efficiency and hardware metrics. The RISC-V ISA enables designers to create custom instructions which are targeted at critical algorithms such as DSP (Digital Signal Processor), artificial intelligence, or cryptography.

2.4. Proposed Lightweight Cryptographic Architectures. IoT is extremely insecure and presents an enormous threat to us. For this reason, in designing lightweight cipher blocks for IoT devices, it is important to recognize that we are building a block cipher that requires a good security level with high throughput speed, while keeping in mind a moderate cost of area and low power consumption.

In this work, the main objective is to propose an enhanced processor equipped with a full hardware security solution. New proposed unrolled architectures of PRESENT and PRINCE ciphers with a 64-bit data size and 128-bit key length are proposed. The two elaborated cipher designs are based on two different techniques: substitution–permutation based network (SPN) and FX-construction. A 32-bit data width of the proposed cipher designs is also elaborated to fulfill ReonV processor datapath. As shown in Table 2, lightweight cryptography block ciphers specifications are presented, including the block/key size (bit), datapath, and the number of rounds.

Table 2: Specifications of lightweight cryptography ciphers.

Cipher	Block size (bits)	Key size (bits)	Characteristics	Datapath (bits)	Round T
PRESENT	64	128	4-bit S-box	32	31
PRINCE	64	128	S-box and matrix layer	32	12

3. PRESENT Cipher Architecture

PRESENT algorithm is a lightweight encryption cipher proposed in 2007 by Bogdanov et al. [21]. It is the most widely used of the light ciphers, and it is part of the ISO/IEC 29192 [22]. For a block length of 64 bits, two key lengths are supported: 80 and 128 bits [23]. PRESENT is a substitution–permutation based network (SPN), which consists of 25 or 31 turns according to the key size. PRESENT cipher used a bit-oriented permutation; this makes it different than other SPN ciphers. Each turn is successively composed of addRoundKey, a parallel 4-bit S-box operation completed by a permutation of the bits as shown in Figure 2:

(i) AddRoundKey: this operation combines the subkeys of each turn with the data, using the eXclusive OR (XOR) bit by bit.

(ii) SBoxLayer: a 4-bit to 4-bit S-box is used. The action of this box is described by the S-box table presented in [21].

(iii) P-Layer: the P-Layer function consists of rearranging the 64 bits in the order as described in [21]. The bit i of state is adjusted to bit position $P(i)$.

(iv) The key schedule: two different key sizes of 80 or 128 bits are allowed to encrypt data. The key update process is varied depending on its size.

Applications such as IoT require high security levels. As a consequence, PRESENT cipher with 128 bit key size security will be adequate to secure communications. Furthermore, a large amount of data will be treated and transmitted; a cipher design with a high throughput and good efficiency is required. A new design will be elaborated without excluding area cost, power, and energy consumption. Encryption and decryption using PRESENT cipher have approximately the same physical requirements.

3.1. PRINCE Cipher Architecture. PRINCE block cipher is the outcome of a collaboration between the Technical University Denmark, NXP Semiconductors, and the Ruhr University Bochum. PRINCE cipher [24, 25] is a lightweight encryption algorithm targeting low latency, unrolled hardware implementations presented in 2012. The key size is about 128 bits, with a block size of 64 bits. Two possible numbers of iterations are possible (11–13) which make it suitable for hardware implementation to reduce latency and achieve better performance. This makes it comparable to AES cipher for 10 rounds using a 128-bit key. Furthermore, the PRINCE rounds are much less complex; this makes it an interesting lightweight block cipher with possible resources optimization and cost reduction [26].

The PRINCE architecture is described by Figure 3. Each round of PRINCE core is categorized as an SPN cipher composed of a key addition, S-box-layer, a linear layer, and a round constant addition. The key is split into two 64-bit keys K_0 and K_0'. The input (Plaintext) is XORed with K_0; then it is processed by a core function using K_1. The output of the core function is XORed by K_0' to produce the final output (Ciphertext):

(i) RCi-add: in this step a 64-bit round constant is XORed with the 64-bit subkey. The values of RCi are presented by Borghoff and Canteaut [24].

(ii) S-Layer: one 4-bit S-box is used. The action of the S-box in hexadecimal notation is presented in [24].

(iii) The Matrices (M/M′-layer): in this layer, a 64-bit state is multiplied, respectively, with a 64×64 matrix M and M'. The M layer (Mult-by-M) is only used in the middle round. The M' layer (Mult-by-M') is used to ensure the α-reflection property ($\mathrm{RC}_i \oplus \mathrm{RC}_{11-i} = \alpha$). The M' mapping is combined in the round functions with a matrix shift row (SR) ($M = \mathrm{SR}\,[\text{Mult-by-}M']$) to guarantee a full diffusion.

Compared to most of lightweight ciphers, PRINCE algorithm has a small number of rounds and the layers constituting a round have low logic depth. As a result, fully unrolled design can reach higher frequencies.

Besides that, the PRINCE cipher is based on FX-construction, which increases the security of a core block cipher. Any claim is noticed regarding PRINCE cipher security against known attacks.

4. Extension of Lightweight Cryptographic Instructions from ReonV Processor

In this part, we will present the principle of integrating new instructions into the ReonV processor core, in order to support the lightweight cryptographic primitives previously developed.

4.1. Integration of Lightweight Cryptoprocessor in the Integer Unit. To add lightweight cryptographic algorithms as extensions for the ReonV processor [17], we are going to build a cryptoprocessor that will be integrated into the integer unit (IU) of the processor, specifically, in the "EXECUTE" stage of the pipeline. The integration of the proposed lightweight cryptoprocessor is shown in the diagram below (Figure 4).

4.2. Lightweight Cryptographic Instruction Configuration Register. In order to know the cryptographic instruction that will be executed, a register called "cryptographic instruction configuration register" must be added on the

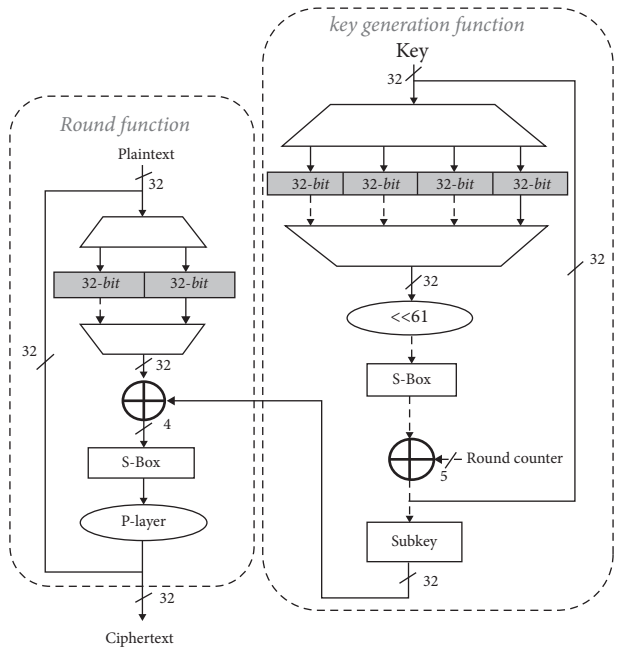

Figure 2: Novel 32-bit datapath for PRESENT hardware implementation.

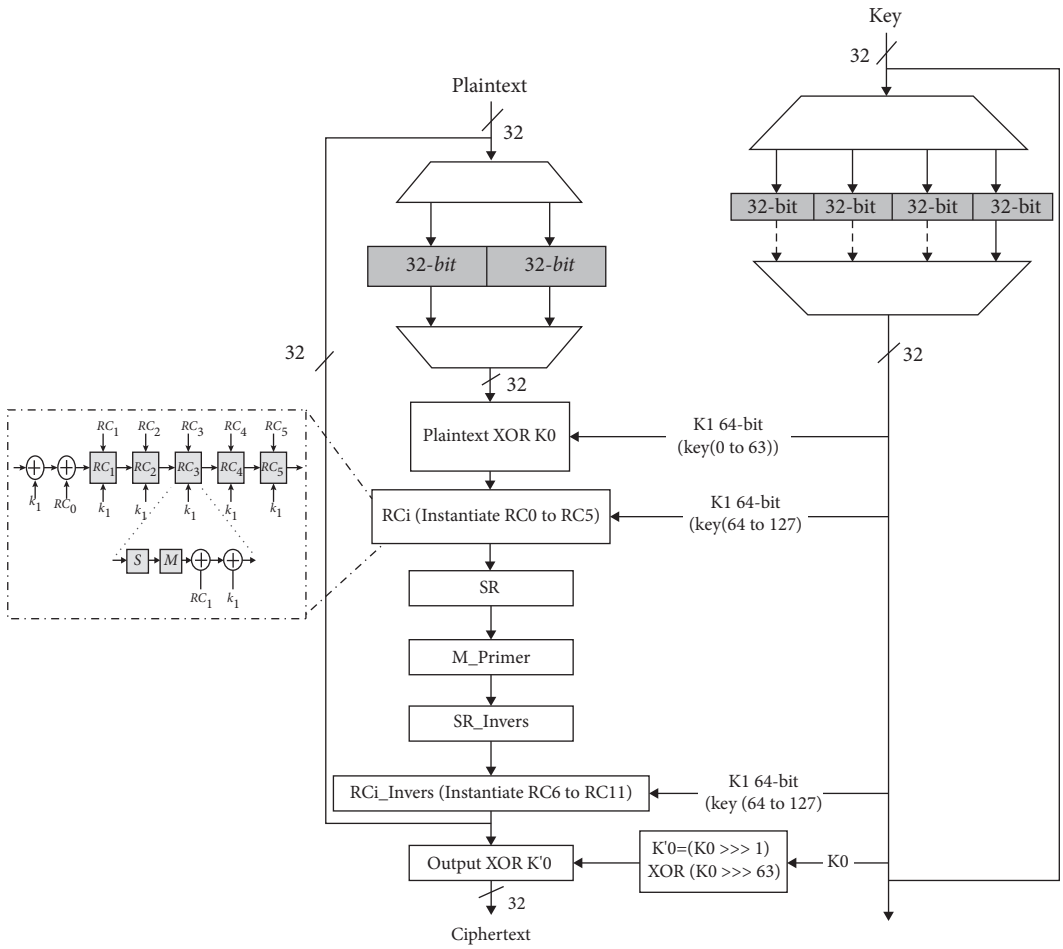

Figure 3: Novel 32-bit datapath for PRINCE hardware implementation.

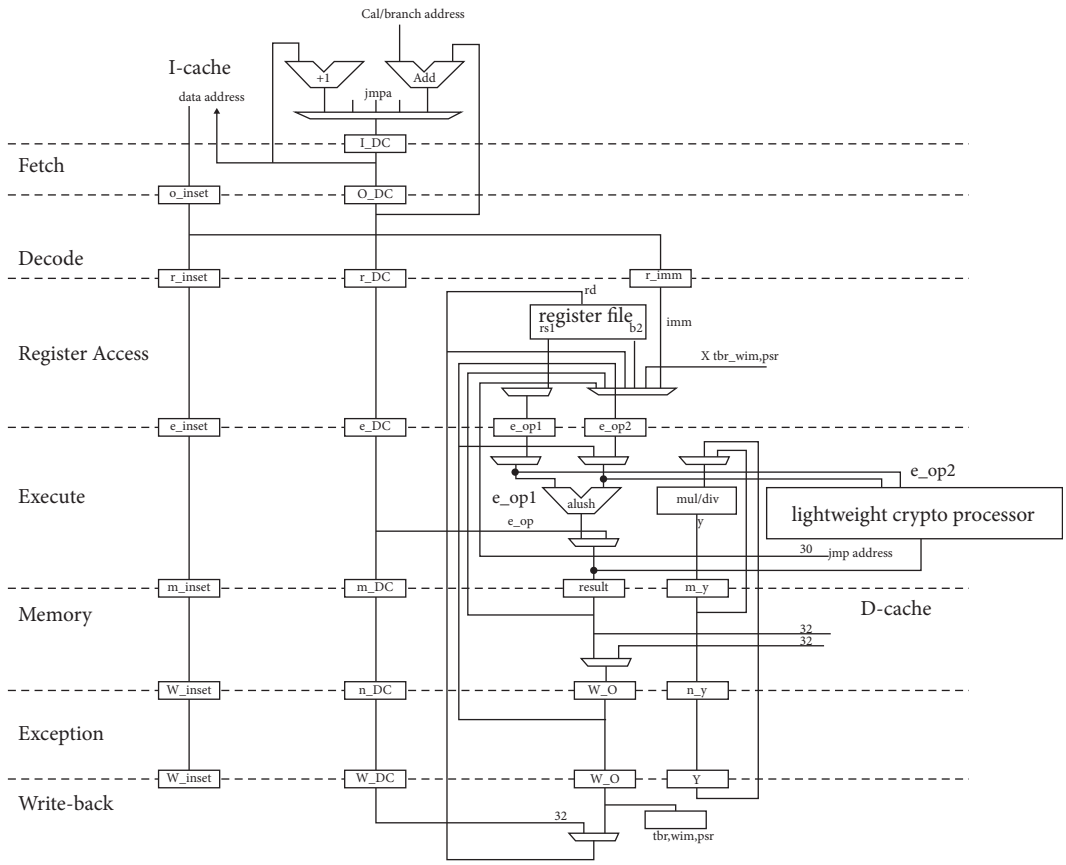

Figure 4: Diagram of the modified 7-stage pipeline of ReonV processor with lightweight cryptoprocessor.

AMBA APB bus (Figure 5). This register contains information on the extension of the cryptographic instructions implemented in the ReonV. It allows you to select the type of instruction that will be executed; the call of these instructions can be selected independently.

These are instructions in different formats; in order to integrate the lightweight cryptographic instructions we have worked on format 3 (FMT3). This register contains the possible values for format 3 of the cryptographic instructions. The « FMT3-OP3-2C » format defined by bits 15, 16, and 17 contains the cryptographic instruction of PRINCE cipher. For "FMT3-OP3-2D" format, defined by bits 12, 13, and 14, it contains the instruction of the algorithm PRESENT. Two versions of ReonV are possible « ISEC-Reonv-Base-Version » which is a basic version and a second extended version « ISEC-Reonv-EXT-Version » (see Figure 6).

The cryptographic instructions of the lightweight algorithms named PRESENT and PRINCE are grouped in the crypto-config register according to the family; in bits 15, 16, and 17 we have added the PRINCE instruction and in the bits 13, 14, and 15 the instruction of the PRESENT cipher is shown in Table 3.

4.3. Instruction Decoding Register. To integrate our two proposed ciphers as lightweight cryptographic instructions, we worked on the instruction decoding register. This register allows instruction decoding. In fact, the two bits 30 and 31 are used to select the form of the instruction. To execute an instruction which belongs to a specified format, it is necessary to load [24 : 19] bits of the decode register as described in Figure 7.

For an instruction belonging to "FMT3_OP3_2C," the decoding register must be filled with the word 0X81600000. In the other side an instruction belonging to the "FMT3_OP3_2D" format, the instruction decoding register, must be loaded with the word 0X81680000.

4.4. Proposed Lightweight Cryptographic Instructions. This section introduced the description of the overloaded lightweight cryptographic instructions (PRESENT and PRINCE), described for RISC-V architecture. A unique format "*f*" is proposed for the elaborated as follows:

$$f \cdot op1, \cdot op2, \cdot r\ d. \tag{1}$$

These instructions have two operands, "Op1" and "Op2." These two registers, predefined by the core of the ReonV processor, are 32 bits in size. The values of the calculation parameters are entered in 32-bit blocks. Thus, to enter operand "Op1" of 128 bits (key), 4 blocks of size 32 bits are needed and 2 blocks for 64-bit operand "Op2" (plaintext). The result of the instruction is stored in the 32-bit destination register "rd" (ciphertext). The "Crypt_en" selects to enable or disable the cipher module (see Figure 8).

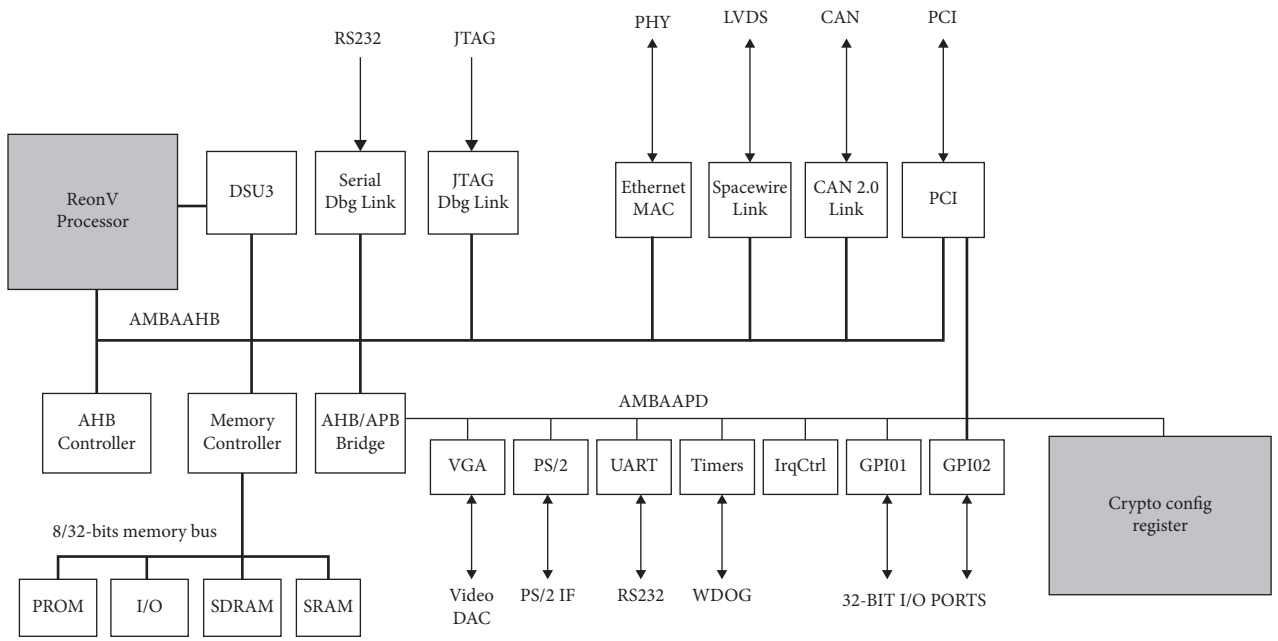

Figure 5: ReonV SOC structure with added crypto-config register.

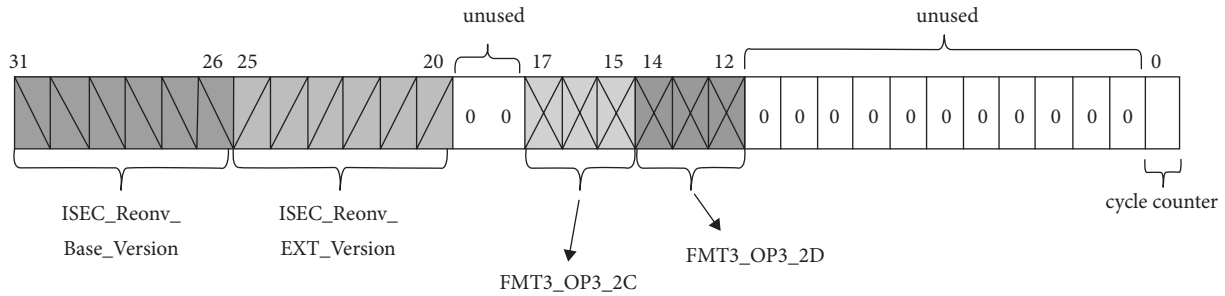

Figure 6: Lightweight cryptographic instruction configuration register.

Table 3: Configuration information for format FMT3.

Value (binary)	Description
FMT3: OP3 = 0x2C	
000	No instruction implemented
001	PRINCE algorithm instruction
FMT3: OP3 = 0x2D	
000	No instruction implemented
001	PRESENT algorithm instruction

The proposed instructions are implemented with the format FMT3 opcode 0x2C and 0x2D; they need 7 clock cycles for 64-bit block cipher.

5. Experimental Results and Security Analysis

In this section, we give the performances overview as well as security analysis of our proposed lightweight cryptographic cipher designs. The hardware implementation of enhanced ReonV processor is evaluated in this section too. All hardware features were derived directly or partly from the results obtained by full design implementation of our VHDL code using Mentor Graphics ModelSim 6.6d and Xilinx Vivado 2021.2 [27]. After place and route, the elaborated hardware implementations have been tested. All the results have been generated for different Xilinx FPGAs Platforms.

5.1. Hardware Implementation of Our Proposed Lightweight Cipher Architectures on FPGA Platforms. The main objective of the proposed lightweight cryptographic cipher designs is to achieve an economic hardware implementation with optimized

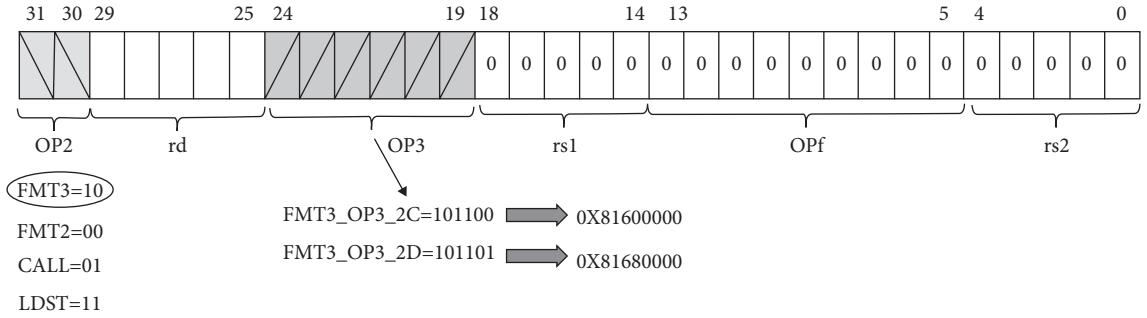

Figure 7: Lightweight cryptographic instruction decoding register.

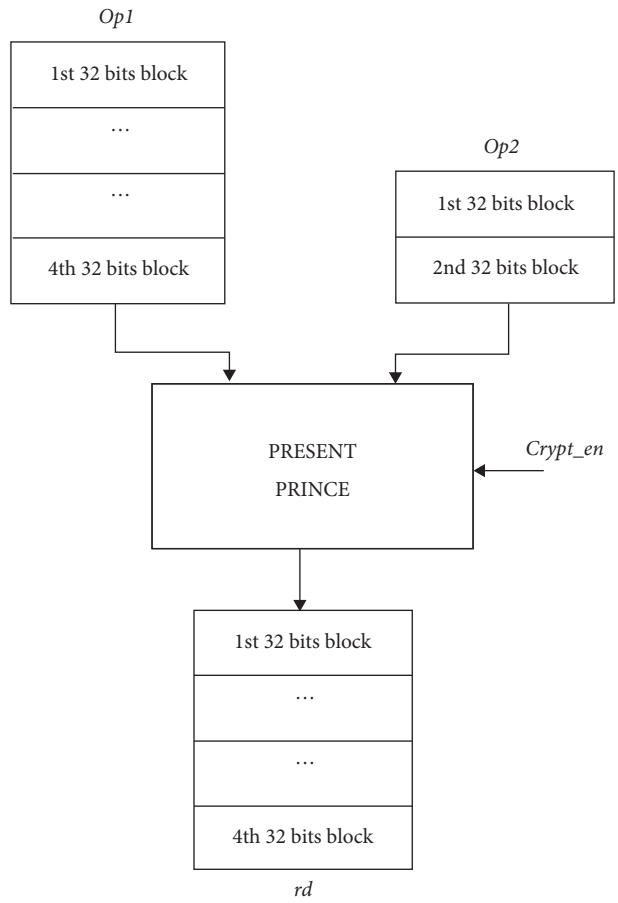

Figure 8: Lightweight cryptographic instructions format for the ReonV architecture.

performances, as well as high security level. The comparison of all the metrics of our proposed lightweight ciphers architectures is shown in Table 4. The plaintext and key loading phases are considered in this work in contrary to other publications.

The comparison of our proposed cipher designs with other published works is taken as reference metrics: the occupied standard reconfigurable logic (equivalent slices) of FPGAs, the running speed, analyzed in terms of clock cycle number, running frequency, and the achieved throughput speed, as well as throughput on area ratio (efficiency), power, and energy per bit consumption. Some metrics are deducted from results published in literature.

A fair comparison with the state of the art is very difficult to achieve due to different FPGA technologies and implementation strategies being used. For this reason, three Xilinx FPGA platforms were chosen as target devices in this work due to their similarities with those used in the literature: Zync-7000, Kintex-7, and Spartan-6 [27]. Furthermore, almost in all related state-of-the-art articles only 64-bit data width is suggested; in this paper both 32-bit and 64-bit data width designs are elaborated for cipher designs. Regarding the total size of input data 64 bit and 128 bit representing, respectively, the plaintext and the secret key, and taking into consideration the internal communication 32-bit datapath

Table 4: Comparison of our proposed hardware implementation results obtained on FPGAs platforms.

Cipher	Designs	FPGA boards	Data width size (bit)	Area (slices)	Latency (cycles)	Speed Max. freq. (MHz)	Throughput (Mbps)	Efficiency (Mbps/slice)	Power (W)	Energy/bit (pJ/bit)
PRESENT	This work	Zync-7000	32	616	7	26.21	239.63	0.389	0.22	918.065
	This work	Kintex-7	32	610	7	62.206	568.740	0.932	0.114	200.443
			64	950	1	39.26	2513	4.223	0.098	39.003
	[28]		64	151	—	100	206.45	1.367	—	—
	This work	Spartan-6	32	594	7	27.62	252.53	0.425	0.11	435.56
	This work		64	858	1	17.88	1144.32	1.334	0.069	60.3
	[29]		64	121	—	13.56	3.47	0.0287	—	—
	C4 [30]		64	474	396	13.56	2.19	0.0046	23.45	10.7×10^{6}
PRINCE	This work	Zync-7000	32	406	7	86.71	792.78	1.952	0.23	290.119
	This work	Kintex-7	32	262	7	94.451	863.552	3.296	0.085	98.431
			64	533	1	78.08	4997	9.375	0.174	34.82
	[31]		64	539	1	61.42	3931	7.290	0.045	11.448
	This work	Spartan-6	32	383	7	38.69	353.70	0.923	0.12	339.235
	This work		64	465	1	34.04	2178.56	4.685	0.115	52.785
	[32]	Virtex-6	64	482	1	65.381	4184	8.68	2.875	687.078
	[33]	Virtex-403	64	956	1	31.76	2032	2.125	0.165	81.175

of the host processor, input/output buffers are used to load the total data size. By this way, the input as well as output data blocks are divided into 32-bit word wide. Therefore, the number of cycles increased by 4 cycles for input block and 2 cycles for output block. This will have a direct impact on the latency, throughput, area occupation, and energy consumed by the design.

To be fair, our proposed 64-bit data width will be compared with identical data sizes of the state of the art.

Various hardware implementations of lightweight cryptographic block ciphers are reported in literature which could be compared to this work as presented in Table 4. Authors in [28, 29] described two hardware implementations of PRESENT cipher. Dalmasso et al. [28] describe a hardware design of PRESENT-128 on Xilinx Kintex-7 FPGA. A normalized frequency of 100 MHz is adopted. They claim that their obtained results show the expected benefits in terms of throughput and area, which allows selecting the best lightweight crypto-ciphers depending on the target device or application.

Compared to our proposed design, in [28] the PRESENT cipher implementation is iterative, and the 31 rounds are performed serially to provide the ciphertext. They implemented a PRESENT cipher architecture with 80-bit key size. In our case, fully unrolled PRESENT cipher architectures with 128-bit key size are elaborated; the 31 rounds are computed in parallel in a single cycle. As can be noticed, our proposed PRESENT cipher design achieves an increase of more than 12x the performances illustrated in [28] for a 64-bit data width on Kintex-7 board. This gain cannot be performed without a loss in consumed resources. The increasing cost in area is about 6x the number of total slices on kintex-7; this is due to the 128-bit key length used. However, we notice a better efficiency in terms of throughput on area ratio (throughput per slice) on Kintex-7 board.

Lara-Nino et al. [29] present an FPGA-based design for PRESENT lightweight block cipher and its implementation results. Although their proposed design allows optimization in area, the throughput is 330x slower than our noticed performance and the throughput on area ratio is 46x worse than our presented solution on Spartan-6 FPGA board.

Neither power consumption nor energy per bit is treated in [28, 29] to be compared with.

In another work [30], Lara-Nino et al. described a PRESENT cipher design implemented on a Spartan-6 FPGA board. We notice an area occupation of 474 slices, which is better by 384 total slices than our proposed design. However, the presented results show a frequency of 13.56 MHz, a latency of 396 cycles with a very low throughput of about 2.19 Mbps, high-power consumption of 23.45 W, and a high energy per bit dissipation. As can be concluded, our proposed PRESENT cipher design is by far better compared to [30] in terms of throughput, throughput on area ratio, dynamic power, and energy per bit consumption.

In [31] Abdullah and Obeid introduce a hardware implementation of PRINCE block cipher on Kintex-7 FPGA platform determined by the quantum cryptography protocol BB84. Obtained data indicates a throughput of 3.931 Gbps.

As can be noticed, our proposed PRINCE cipher block is almost 2x better in terms of throughput than result illustrated in [31] when implemented on kintex-7 platform, with a throughput of approximately 5 Gbps and better efficiency of 9.375 Mbps per slice. The noticed area occupation is little better than [31]; however a waste in power and energy per bit consumption is noticed.

In another work [32], obtained results show a throughput of 4.18 Gbps and 2.875 W as power consumption when implemented on Virtex-6 FPGA board. In [33] a PRINCE IP core architecture has been elaborated to encrypt and decrypt the data in only one clock cycle with low latency and high speed. Obtained results with Virtex-403 FPGA board indicates a frequency of 31.76 MHz, a throughput of 2.032 Gbps, and a power consumption of 0.165 W. A comparison with results presented in [32, 33] cannot be continued because of different FPGA platforms.

Comparing the results of our two proposed unrolled cipher designs, PRINCE cipher presents better throughput speed, less area occupation, good throughput per slice ratio, and reduced energy per bit consumption compared with PRESENT cipher on all FPGA boards for a 64-bit data width. This can be explained by the reduced number of rounds for PRINCE cipher (12 rounds) compared to PRESENT cipher (31 rounds). The layers constituting a round of PRINCE have low logic depth and the operation's complexity is much lower. The logic depth of a path represents the number of combinatorial gates between input and output; it will be linked directly to the latency of the circuit. As a result, complex operations will be associated with a large number of combinatorial gates and a reduced throughput speed.

We can also notice that obtained results of our elaborated cipher designs depend on the technology of FPGA platforms used. Indeed, Kintex-7 FPGA is a 28 nm process technology; on the other side the spartan-6 FPGA board is a 45 nm process technology.

Security analysis of our proposed lightweight cipher designs is elaborated in the next section.

5.2. Security Analysis of Our Proposed Lightweight Ciphers Architecture. To test security of our proposed lightweight cryptographic architectures against the most common attacks, statistical analysis as well as key sensibility analysis is achieved.

5.3. Statistical Analysis. In this section, security analysis of the experimental results is discussed. Security analysis covers histogram, entropy, and correlation analysis [34].

5.4. Histograms of Encrypted Images. The histogram shows distribution of pixel intensities of the image. It illustrates how the gray levels of the pixels in an image are distributed. Based on the obtained histogram, an attacker can deduce the plain-pixels by carrying out a frequency analysis (statistical attack). To prevent statistical attack, histogram of cipher-image should statistically have a uniform distribution. Relatively uniform distribution of the cipher-image is a sign of a good image encryption quality [34]. In Figures 9–12, we give the results obtained for Lena, Peppers, Baboon, and Barbara of size 256×256.

As can be concluded from the following figures, the histograms of the ciphered images are very close to the uniform distribution and are completely different from the plain images with good diffusion properties.

5.5. Entropy Analysis. Entropy is a statistical measure of uncertainty or randomness associated with information. It can be used to define the image texture. Thus, entropy can be considered as an assessment criterion for the significance of cipher algorithm [35]. The higher the entropy the better the encryption algorithm. The formula of entropy as given by Shannon is as follows:

$$H(X) = \sum_{i=0}^{255} p(x_i)\, log_2(x_i), \tag{2}$$

where $H(X)$ is the entropy of the encrypted image, and $P(x_i)$ is the probability of each gray level appearance ($x_i = 0, 1, \ldots, 255$). In the case of equal probability levels, the entropy is maximum. The entropy should be ideally equal to 8 in the case of random images.

As presented in Table 5, it can be noticed that the entropy of all ciphered images is close to maximum, depicting an attribute of the algorithm.

From obtained entropy values of ciphered images, we notice that results are very close to 8. The entropy of each ciphered image is greater than 7.90. We conclude that the ciphered images have a good local randomness; hence our proposed lightweight cryptographic architectures can be considered as safe from entropy attack.

6. Correlation Analysis

Correlation analysis is an extensively used technique to decide the strength of a relationship between two pixels in an image [36]. For ciphered image, the less the adjacent pixel correlation will be, the better the encryption process is capable of hiding the details of the original image.

The correlation coefficient is computed using equations (3)–(6) where x and y are the grayscale values of two adjacent pixels of the image in horizontal, vertical, and diagonal directions.

$$\rho(x, y) = \frac{\mathrm{Cov}(x, y)}{\sqrt{D(x).D(y)}}, \tag{3}$$

where

$$\mathrm{Cov}(x, y) = \frac{1}{n} \sum_{i=1}^{n} (xi - E(x))(yi - E(y)), \tag{4}$$

$$D(x) = \frac{1}{n} \sum_{i=1}^{n} (xi - E(x))^2, \tag{5}$$

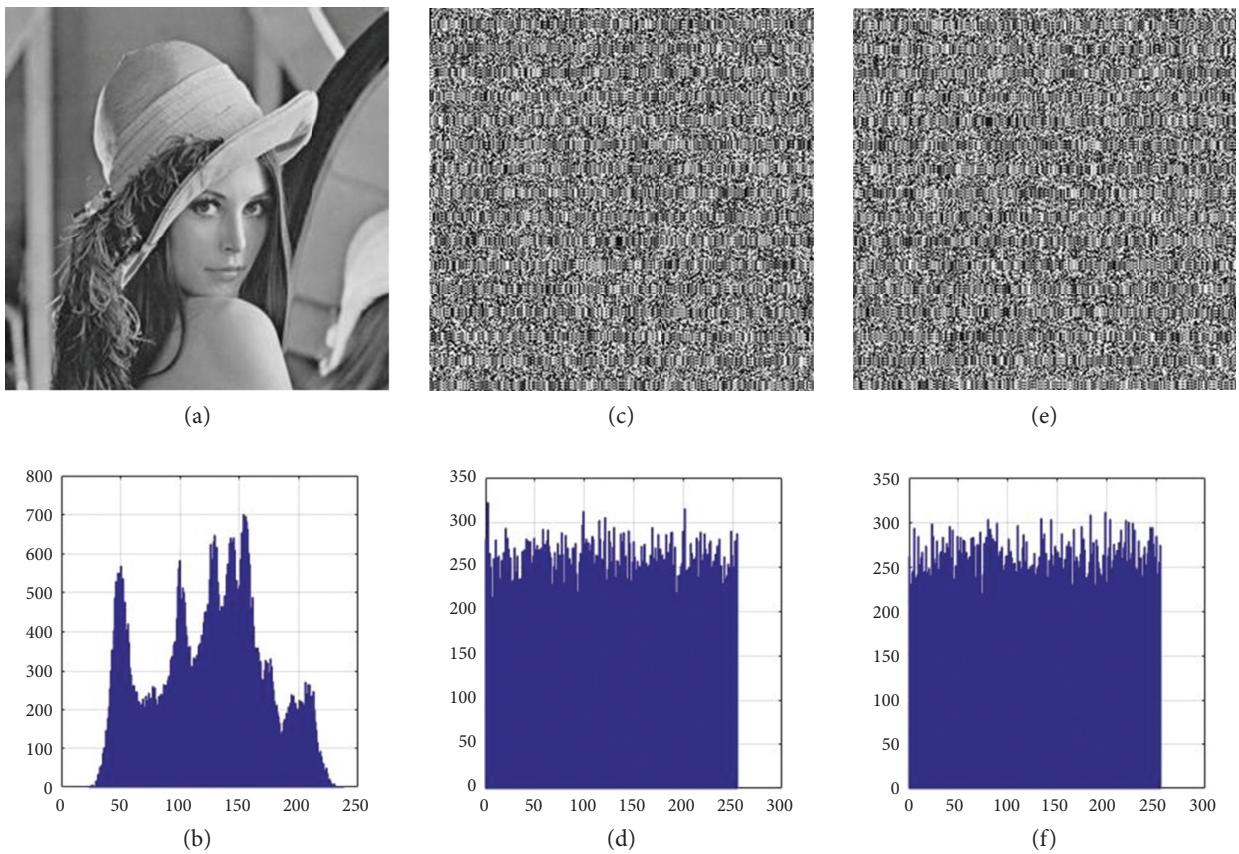

Figure 9: Result of Lena image. (a) Original image. (b) Histogram of the original image. (c) PRESENT ciphered. (d) Histogram of PRESENT ciphered. (e) PRINCE ciphered. (f) Histogram of PRINCE ciphered.

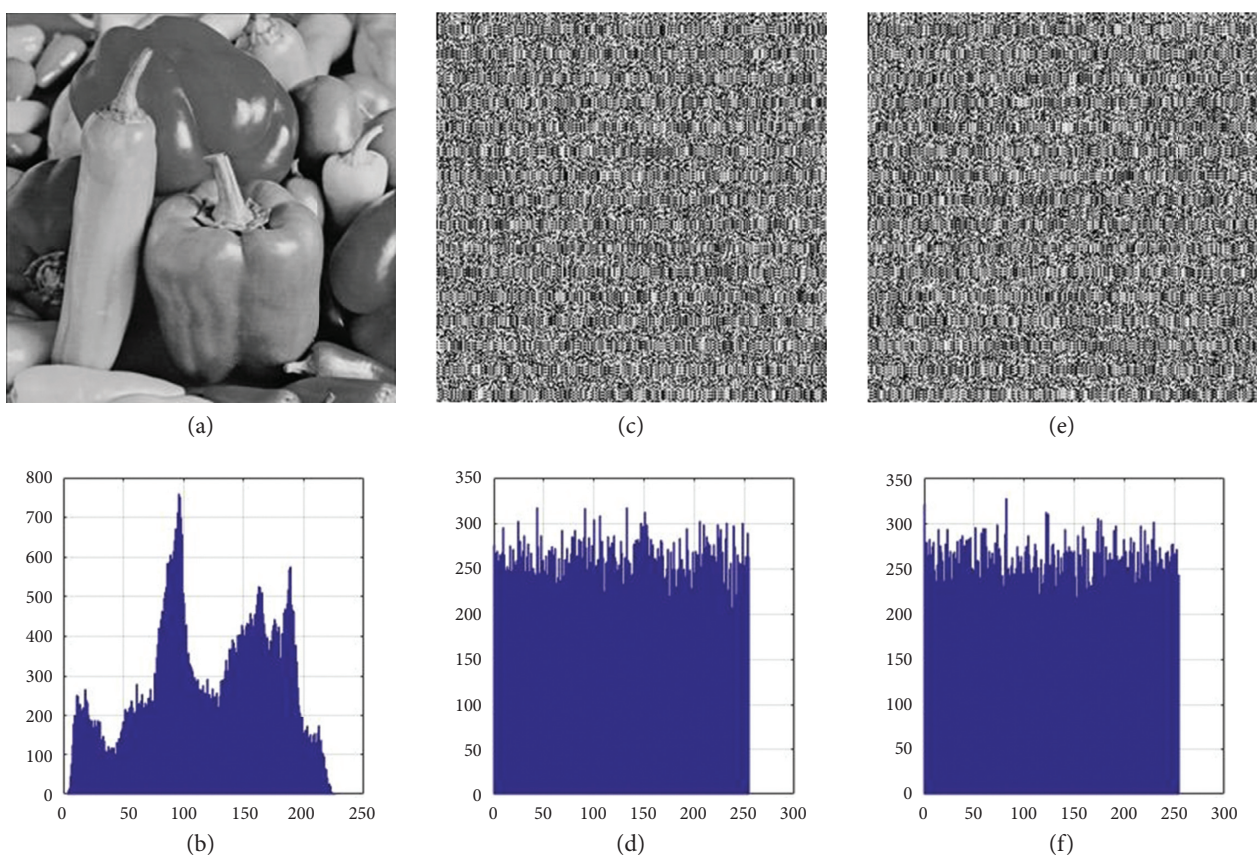

Figure 10: Result of Pepper image. (a) Original image. (b) Histogram of the original image. (c) PRESENT ciphered. (d) Histogram of PRESENT ciphered. (e) PRINCE ciphered. (f) Histogram of PRINCE ciphered.

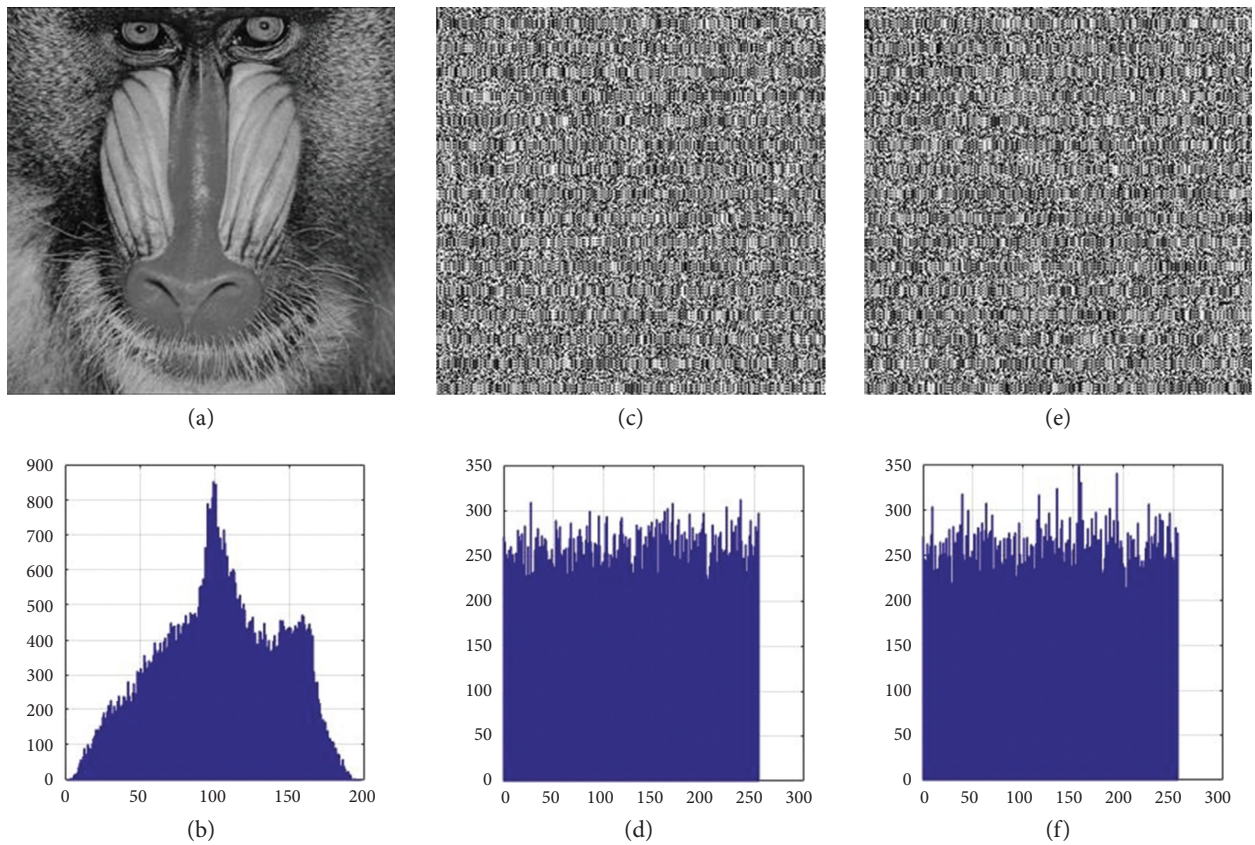

Figure 11: Result of Baboon image. (a) Original image. (b) Histogram of the original image. (c) PRESENT ciphered. (d) Histogram of PRESENT ciphered. (e) PRINCE ciphered. (f) Histogram of PRINCE ciphered.

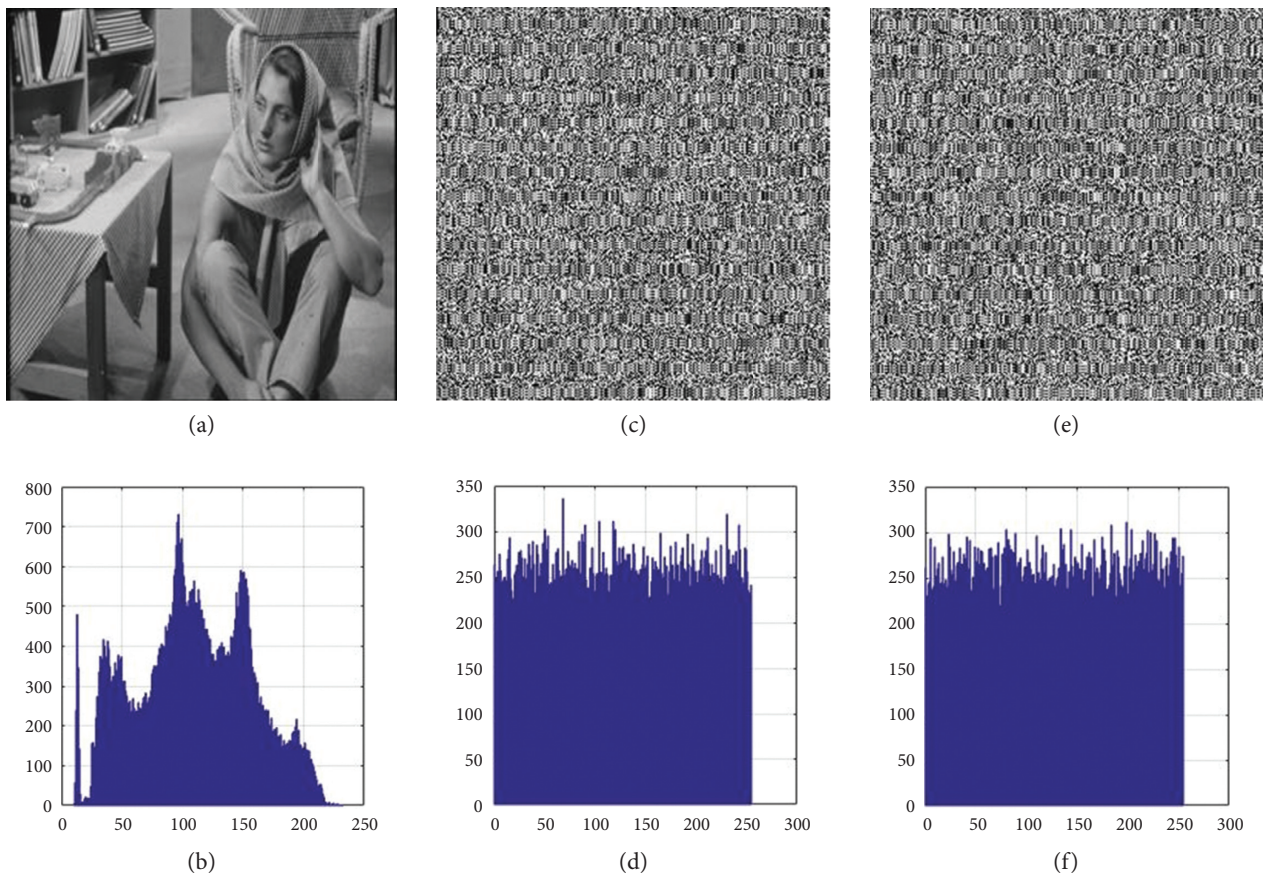

Figure 12: Result of Barbara image. (a) Original image. (b) Histogram of the original image. (c) PRESENT ciphered. (d) Histogram of PRESENT ciphered. (e) PRINCE ciphered. (f) Histogram of PRINCE ciphered.

Table 5: Entropy results obtained.

Entropy	Lenna	Pepper	Baboon	Barbara
Plain image	7.4440	7.5939	7.3102	7.5199
PRESENT cipher	7.9537	7.9533	7.9526	7.9580
PRINCE cipher	7.9580	7.9552	7.9561	7.9580

$$E(x) = \frac{1}{n}\sum_{i=1}^{n} xi. \tag{6}$$

For a plain image, two adjacent pixels have a strong linear relationship and the correlation coefficient is close to 1. For a good ciphered image, the correlation should be as low as possible (close to zero) between adjacent pixels.

Table 6 shows that correlation coefficients values are very close to zero between adjacent pixels. There is no correlation between the plain and ciphered images. As a result, there is any similarity between plain and encrypted images, proving a very good achieved confusion by the proposed lightweight ciphers. All of them present good ability to resist against statistical attacks.

6.1. Key Sensitivity Analysis. A secure cipher design should be very sensitive to the secret key; to measure the key sensitivity of our proposed lightweight cipher designs, we randomly choose two secret keys with only one-bit (LSB) difference to encrypt the plain image. The key sensitivity evaluation is based on two parameters: The Number of Pixels Change Rate (NPCR) and the Unified Average Changing Intensity (UACI). The two parameters NPCR and UACI are calculated according to the following equations [38]:

$$\begin{aligned}
\text{NPCR} &= \frac{1}{M * N}\sum_{i,j} D(i,j) * 100\%,\\
D(i,j) &= \begin{cases} 1 \ if \ C_1(i,j) \neq C_2(i,j) \\ 0 \ if \ C_1(i,j) = C_2(i,j) \end{cases},\\
\text{UACI} &= \frac{1}{M * N * 255}\sum_{i,j}\left|C_1(i,j) - C_2(i,j)\right| * 100\%.
\end{aligned} \tag{7}$$

In the previous equations, i and j are the row and column indexes of the image, respectively. M and N are, respectively, the length and width sizes of ciphered images C_1 and C_2.

The values of NPCR and UACI demonstrate that the algorithm is robust against differential attack. As can be concluded from Table 7, the average obtained values of NPCR and UACI are close to the optimal values of NPCR and UACI which are 99.609% and 33.463%, respectively. Therefore, our proposed cipher designs are highly sensitive to small changes in the secret key and resistant against differential attacks.

6.2. Comparison with Existing Cryptographic Solutions. In this section, we compare our proposed cipher designs with several cryptographic solutions. In the comparison, we focus on the statistical analysis and key sensitivity with previous works. Table 8 presents a comparison of entropy and correlation coefficients for the proposed designs and other works from three directions for the image Lena of size 256×256.

Table 6: Correlation coefficient of two adjacent pixels of plain and ciphered images.

Image		Horizontal	Vertical	Diagonal
Lena	Plain image	0.8857	0.9517	0.8721
	PRESENT ciphered	$-9.8017e-04$	−0.0119	−0.0012
	PRINCE ciphered	−0.0072	−0.0016	−0.0017
Pepper	Plain image	0.9599	0.9679	0.9404
	PRESENT ciphered	−0.0156	−0.0013	$4.2070e-04$
	PRINCE ciphered	−0.0013	−0.0111	0.0063
Baboon	Plain image	0.8778	0.8342	0.7881
	PRESENT ciphered	−0.0100	−0.0016	0.0076
	PRINCE ciphered	0.0123	0.0039	0.0094
Barbara	Plain image	0.9078	0.9461	0.8835
	PRESENT ciphered	−0.0096	−0.0167	0.0022
	PRINCE ciphered	−0.0072	−0.0016	−0.0017

We can see that our proposed lightweight cipher designs have competitive performances compared with other existing cryptographic schemes. Our proposed cipher designs reveal the randomness and the degree of ambiguity in the ciphered image.

In order to compare the key sensitivities for our two elaborated cipher designs, the image Lena is encrypted by the secret key and then by changing a single bit in the original secret key. Both NPCR and UACI of encrypted images are calculated as listed in Table 9.

The test results support that the approach we proposed has at least the same excellent performance as the other proposed solution. Therefore, the cipher designs we suggest can resist differential attacks effectively.

6.3. Hardware Implementation of Enhanced ReonV Processor on FPGA. We have synthesized the customized ReonV processor used to optimize the lightweight cryptographic instructions with Spartan-6 FPGA Board (Table 10).

When integrated in the integer unit stage of the processor, we can conclude that the area occupation increases a little by 1% for the two proposed instructions in terms of

Table 7: NPCR and UACI key sensitivity tests.

Cipher	Image	NPCR (%)	UACI (%)
PRESENT	Baboon	99.5316	33.3880
	Barbara	99.5636	33.3658
	Lena	99.5300	33.5705
	Pepper	99.6033	33.4735
PRINCE	Baboon	99.6017	33.7597
	Barbara	99.5911	33.6328
	Lena	99.5239	33.5062
	Pepper	99.5605	34.0071

Table 8: Comparison on statistical analysis.

Cipher	Entropy	Correlation		
		Horizontal	Vertical	Diagonal
Proposed PRESENT	7.9537	$-9.8017e-04$	−0.0119	−0.0012
Proposed PRINCE	7.9580	−0.0072	−0.0016	−0.0017
[39]	7.9565	−0.009072	0.016579	0.000584
[39]	7.9562	0.000338	−0.012928	−0.003296
[39]	7.9549	−0.009522	−0.007509	0.001907
[40]	7.9995	−0.0237	−0.00178	−0.0284
[41]	7.9967	0.00199	0.00354	0.00265
[42]	7.9993	0.0012	0.0003	−0.0010
[43]	7.9976	0.0023	0.0158	0.0147

Table 9: Comparison on key sensitivity.

Cipher	NPCR (%)	UACI (%)
PRESENT	99.5300	33.5705
PRINCE	99.5239	33.5062
[40]	99.6221	33.5887
[41]	99.6307	33.3619
[42]	99.5964	33.4762
[43]	99.6101	33.4583

Table 10: Synthesis results of customized ReonV processor with Spartan-6 FPGA board.

Design	Area (resources)		Performance	
	Occupation (slice LUTs + FF-pairs)	% AreaOverhead	Max. freq. (MHz)	Throughput (Mbps)
Original ReonV without ext.	3757	—	133.074	—
Modified ReonV with PRESENT instr.	3770	1%	132.769	1213.888
Modified ReonV with PRINCE instr.	3765	1%	132.103	1207.798

Slice LUTs plus FF-pairs. On the other side, we noticed the resulting improvement in the throughput is significant for both PRESENT and PRINCE instructions by 480% and 341%, respectively. Furthermore, the maximum clock rates reported by the synthesis tools are 133.074 MHz for the original ReonV processor.

In case of enhanced processor version, the maximum frequency reaches 132.769 MHz and 132.103 MHz for the modified one with PRESENT and PRINCE instructions, respectively. The difference is marginal, and the overall implementation can be optimized.

7. Conclusion

To ensure IoT devices, security must extend beyond software-based security into hardware-based security. In this work, an efficient and secure support of cryptography on embedded processors is elaborated. We have focused on the

promising approach of instruction set extensions. Two instruction set extensions for 32-bit processor for PRESENT and PRINCE lightweight cryptographic ciphers are described. The designs and implementations of our proposed 32-bit data width ciphers are detailed. We noticed that our proposed block ciphers present high throughputs as well as good efficiency with moderate power consumption and energy dissipation. Security level is analyzed to prove the robustness of these designs. The results show that our proposed lightweight cryptographic modules are secure enough against possible attacks.

The proposed lightweight cryptographic ciphers are integrated directly into the core of the processor, which makes them available by means of additional custom instructions. With hardware costs of about 4K Slice LUTs plus FF-pairs with an area overhead about 1%, a throughput of 1.2 Gbps is reached. The main advantage is that a single instruction is needed to implement a full lightweight cryptographic instruction which can be instantiated in software routines exactly as the processor's base instructions with high security level.

Data Availability

The experimental data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] D. Bandyopadhyay and J. Sen, "Internet of Things: applications and challenges in technology and standardization," *Wireless Personal Communications*, vol. 58, no. 1, pp. 49–69, 2011.

[2] H. Sundmaeker, P. Guillemin, P. Friess, and S. Woelfflé, "Vision and challenges for realising the internet of Things," *cluster of European research projects on the internet of things—CERP IoT*, 2010.

[3] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): a vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[4] A. M. Kerry, B. Larry, S. T. Meltem, and M. Nicky, "Report on lightweight cryptography," *National Institute of Standards and Technology Internal Report*, vol. 8114, March 2017.

[5] B. Kim, J. Cho, B. Choi, J. Park, and H. Seo, "Compact implementations of HIGHT block cipher on IoT platforms," *Security and Communication Networks*, vol. 201910 pages, 2019.

[6] A. Varici, G. Saglam, and A. Aysu, "Fast and efficient implementation of lightweight crypto algorithm PRESENT on FPGA through processor instruction set extension," in *Proceedings of the 2019 IEEE east-west design & test symposium (EWDTS), batumi*, pp. 1–5, Georgia, November 2019.

[7] G. H. Eisenkraemer, F. G. Moraes, L. L. de Oliveira, and E. Carara, "Lightweight Cryptographic Instruction Set Extension on Xtensa Processor," in *Proceedings of the 2020 IEEE international symposium on circuits and systems (ISCAS)*, pp. 1–5, Seville, Spain, October 2020.

[8] B. Marshall, G. R. Newell, D. Page, M.-J. O. S. Saarinen, and C. Wolf, "The design of scalar AES instruction set extensions for RISC-V," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2021, no. 1, pp. 109–136, 2020.

[9] M. Werner, T. Unterluggauer, D. Schaffenrath, and S. Mangard, "Sponge-based control-flow protection for IoT devices," in *Proceedings of the 2018 IEEE European symposium on security and privacy (EuroS&P)*, vol. 2018, pp. 214–226, London, UK, April 2018.

[10] P. Grabher, J. Großschädl, and D. Page, "Light-weight instruction set extensions for bit-sliced cryptography," in *Cryptographic Hardware and Embedded Systems – CHES 2008*, E. Oswald and P. Rohatgi, Eds., vol. 5154, Berlin, Heidelberg, Springer, 2008.

[11] T. Yalcin, B. Bilgin, and E. Kavun, "Towards an ultra lightweight crypto processor," in *Proceedings of the in 2011 workshop on lightweight security & privacy: Devices, Protocols, and Applications*, pp. 76–83, Istanbul, Turkey, 2011.

[12] Y. Liu, R. Cheung, and H. Wong, "Lightweight secure processor prototype on FPGA," in *Proceedings of the 2018 28th international conference on field programmable logic and applications (FPL)*, pp. 443–4431, Dublin, August 2018.

[13] S. Steinegger and R. Primas, "A fast and compact RISC-V accelerator for ascon and friends," in *Proceedings of the 19th smart card research and advanced applications conference*, January 2021.

[14] T. Park, H. Seo, S. Lee, and H. Kim, "Secure data encryption for cloud-based human care services," *Journal of Sensors*, vol. 2018, pp. 1–10, 2018.

[15] N. X. P. Semiconductors, "LPC55S6x: high efficiency Arm® cortex®-M33-based microcontroller family, product data sheet," *Rev. 2.1 —*, vol. 9, 2020.

[16] Leon3, "LEON3 Processor," 2021, https://www.gaisler.com/index.php/products/processors/leon3.

[17] K. Asanović and D. A. Patterson, "Instruction sets should be free: the case for RISC-V. Technical Report UCB/EECS-2014-146," 2014, http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-146.html.

[18] I. Taştan, M. Karaca, and A. Yurdakul, "Approximate CPU design for IoT end-devices with learning capabilities," *Electronics*, vol. 9, no. 1, 2020.

[19] K. Asanović and D. A. Patterson, "Instruction sets should Be free: the case for RISC-V," *technical report No. UCB/EECS-2014-146, EECS Department*, University of California, Berkeley, 2014.

[20] V. Reon, "lcbcFoo," 2018, https://github.com/lcbcFoo/ReonV.

[21] A. Bogdanov, L. Knudsen, G. Leander, C. Paar, A. Poschmann, and M. Robshaw, "Present — an ultra-lightweight block cipher," in *Proceedings of Workshop on Cryptographic Hardware and Embedded Systems—CHES 2007*, vol. 4727, pp. 450–466, Springer-Verlag, Vienna, Austria., 2007.

[22] Iso/Iec 29192-1, "Lightweight cryptography," *Information technology — Security techniques — Lightweight cryptography — Part 1: Gene*, 2019.

[23] C. Lee, "Biclique cryptanalysis of PRESENT-80 and PRESENT-128," *The Journal of Supercomputing*, vol. 70, no. 1, pp. 95–103, 2014.

[24] J. Borghoff and A. Canteaut, "Prince – a low-latency block cipher for pervasive computing applications," in *Advances in Cryptology – ASIACRYPT*, X. Wang and K. Sako, Eds., vol. 7658, Berlin, Heidelberg, Springer, 2012.

[25] A. Biryukov, *DES-X (or DESX), encyclopedia of cryptography and security*p. 331, 2nd edition, Springer, Boston, MA, 2011.
[26] A. Poschmann, "Lightweight cryptography – cryptographic engineering for a pervasive world," 2009, http://eprint.iacr.org/2009/516.
[27] xin, "xilinx," 2021, https://www.xilinx.com/.
[28] L. Dalmasso, F. Bruguier, P. Benoit, and L. Torres, "Evaluation of SPN-based lightweight crypto-ciphers," *In IEEE Access*, vol. 7, pp. 10559–10567, 2019.
[29] C. A. Lara-Nino, M. Morales-Sandoval, and A. Diaz-Perez, "Novel FPGA-based low-cost hardware architecture for the PRESENT block cipher," *2016 Euromicro Conference on Digital System Design (DSD)*, vol. 2016, pp. 646–650, 2016.
[30] C. A. L. Nino, A. diaz-Perez, and M. Morales-Sandoval, "energy and area costs of lightweight cryptographic algorithms for authenticated encryption in WSN," *Security And Communication Networks*, vol. 2018, Article ID 5087065, 14 pages, 2018.
[31] A. A. Abdullah and N. R. Obeid, "Efficient implementation for PRINCE algorithm in FPGA based on the BB84 protocol," *Journal of Physics: Conference Series*, vol. 1818, no. 1, pp. 012216–6, December 2021.
[32] Y. A. Abbas, R. Jidin, N. Jamil, M. R. Z'aba, M. E. Rusli, and B. Tariq, "Implementation of PRINCE algorithm in FPGA," *Proceedings of the 6th International Conference on Information Technology and Multimedia*, vol. 20144 pages, 2014.
[33] Y. A. Abbas, R. jidin, N. jamil, and M. reza Z`aba, "lightweight PRINCE algorithm IP core for securing GSM messaging using FPGA," *Research Journal of Information Technology*, vol. 8, pp. 17–28, 2016.
[34] Y. Zhang, "Security analysis of a chaos triggered image encryption scheme," *Multimedia Tools and Applications*, vol. 78, no. 22, pp. 31303–31318, 2019.
[35] A. Jolfaei and A. R. Mirghadri, "An image encryption approach using chaos and stream cipher," *Journal of theoretical and applied information technology*, vol. 19, no. 2, pp. 117–125, 2010.
[36] Y. Wu, Y. Zhou, G. Saveriades, S. Agaian, J. P. Noonan, and P. Natarajan, "Local Shannon entropy measure with statistical tests for image randomness," *Information Sciences*, vol. 222, pp. 323–342, 2013.
[37] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, University of Illinois at Urbana-Champaign, Champaign, IL, USA, 2nd edition, 2006.
[38] J. Wu, P. Noonan, and S. Agaian, "NPCR and UACI randomness tests for image encryption, cyber journals: multidisciplinary journals in science and technology," *Journal of Selected Areas in Telecommunications (JSAT)*, vol. 1, no. 2, pp. 31–38, 2011.
[39] W. El, H. Youssef, A. Abdelli, F. dridi, and M. Machhout, "Hardware implementation of secure lightweight cryptographic designs for IoT applications," *Security and Communication Networks*, vol. 2020, Article ID 8860598, 13 pages, 2020.
[40] P. Ramasamy, V. Ranganathan, S. Kadry, R. Damaševičius, and T. Blažauskas, "An image encryption scheme based on block scrambling, modified zigzag transformation and key generation using enhanced logistic—tent map," *Entropy*, vol. 656, p. 21, 2019.
[41] H. Liu, Bo Zhao, J. Zou, L. huang, and Y. Liu, "A lightweight image encryption algorithm based on message passing and chaotic map," *Security and Communication Networks*, vol. 2020, Article ID 7151836, 12 pages, 2020.
[42] H. Umar and A. A. Naveed, "A novel image encryption scheme based on an elliptic curve," *Signal Processing*, vol. 155, pp. 391–402, 2019.
[43] X. Zhang, T. Wu, Y. Wang, L. Jiang, and y. Niu, "A novel chaotic image encryption algorithm based on Latin square and random shift," *Computational Intelligence and Neuroscience*, vol. 2021, Article ID 2091053, 13 pages, 2021.