International Journal of Electrical and Computer Engineering (IJECE)
Vol. 12, No. 3, June 2022, pp. 2194~2205
ISSN: 2088-8708, DOI: 10.11591/ijece.v12i3.pp2194-2205

Lightweight ANU-II block cipher on field programmable gate array

Yousif Nihad Hatif[1], Yasir Amer Abbas[1], Mudhafar Hussein Ali[2]
[1]Department of Computer Engineering, College of Engineering, University of Diyala, Diyala, Iraq
[2]Computer Engineering Department, Engineering College, Al-Iraqia University, Baghdad, Iraq

Article Info

Article history:

Received Jun 30, 2021
Revised Dec 23, 2021
Accepted Jan 2, 2022

Keywords:

ANU-II
Block cipher
Field programmable gate array
Lightweight cryptography
VHDL

ABSTRACT

Nowadays the number of embedded devices communicating over a network is increasing. Thus, the need for security appeared. Considering various constraints for the limited resources devices is very important. These constraints include power, memory, area and latency. A perfect environment for satisfying requirements of security in limited resources devices is lightweight cryptography. A recent lightweight algorithm that has a low area and high throughput which is the ANU-II block cipher. Many technologies like the internet of things (IoT) needed lightweight hardware architectures to provide security for it. In IoT issues like the size of memory, power consumption and smaller gate counts need to take care of by using lightweight cryptography. This paper presents hardware lightweight data path implementation for the ANU-II algorithm using field programmable gate array (FPGA). This paper presents a hardware implementation of a 64-bit ANU-II block cipher. Also, this research presents comparisons based on various design metrics among our data path for the ANU-II cipher and other existing data path designs. The result of the proposed design shows a high throughput of 1502.31, 1951.86, and 2696.47 Mbps. Also, it shows the high efficiency of 7.0201, 31.9977, and 10.6579 Mbps/slice as compared to other ciphers in this paper.

Corresponding Author:

Yousif Nihad Hatif
Department of Computer Engineering, College of Engineering, University of Diyala
Baquba, Diyala, Iraq
Email: yousifnh1989@gmail.com

1. INTRODUCTION

The world now becomes interconnected with many devices through the internet of things (IoT), that do many different tasks, for example, radio frequency identification (RFID), sensors, and smart devices [1], [2]. IoT has interfaces to communicate with information networks therefore the need for security has been appeared to protect the devices and their information from different security attacks. In IoT, the devices have limited resources [3], [4]. These limited resources include computation cost, area, and power consumption. Standard cryptographic algorithms did not take into consideration the limited resources of IoT devices therefore it is unsuitable to work on these devices. So new lightweight cryptography algorithms have been proposed and analyzed to meet the requirements imposed by limited resources devices [5]. Ciuonzo *et al.* [6] address the problem of detection of a non-cooperative target, that emits an unknown signal. Darvishi *et al.* [7] proposed general machine-learning-based architecture to detect anomalies in measurements from sensors and identifying the faulty ones.

ANU-II algorithm is an efficient modified version of the ANU algorithm. To simplify ANU efforts have been made to get efficient design metrics for limited resources devices. Compared to existing lightweight ciphers ANU-II is simple but efficient in all metrics of design. ANU-II is a 25 round block cipher and supports Plaintext of 64-bit and key 80/128-bit [8]. To the best of our knowledge, the ANU-II is not implemented in field programmable gate array (FPGA) before. So in this paper, we have implemented ANU-II on the FPGA platform and compare it with existing implementations of other algorithms.

An iterative round-based architecture for the ultra-lightweight LED block cipher has been designed in [9]. This FPGA architecture focused on 64-bit data size and 64-bit key size. The results showed that this architecture achieves better trade-offs in performance and area than existed designs because it used look-up tables in MixColumns module [9]. Two FPGA structures with low cost and two-cycle have been created for the PRINCE algorithm on Virtex-4 and Virtex-6 [10]. The low-cost architecture is designed based on the lowest number of computation resources. It requires 11 clock cycles to perform the encryption operation for 64-bit data with a 128-bit key. The two-cycle architecture of the PRINCE algorithm is achieved with a processing element which is a reconfigurable element [10]. However, the speed and efficiency of this work are higher than it. Also, the PRINCE algorithm is implemented on FPGA by using the quantum cryptography protocol (BB84) in [11]. Only one clock cycle is required by this architecture for the encryption process [11]. However, the cost is higher than the presented work in this paper for example for Virtex4 the presented work in [11] consumes 1026 slices whereas in this paper for the Virtext4 platform only 253 slices are consumed. Several scalars and pipelined FPGA architectures for the SIMON block cipher were designed by [12]. An emphasis was on energy and power for these two implementations. Although the scalar design consumes fewer resources and less power than pipelined design, the pipelined architecture achieves better throughput and consumes less energy [12].

Two different FPGA architectures are designed for the KLEIN block cipher [13]. The first architecture is the iterative architecture that has a data path of 8-bit. The second architecture is the parallel architecture with a 16-bit data path. This parallel architecture achieves high throughput than iterative architecture [13]. Moreover, different architectures have been designed for many lightweight algorithms using different data paths. Numerous optimization strategies have been employed for advanced encryption standard (AES) 32-bits by Bui *et al.* in [14]. They achieved low power, low cost, and high-throughput design with several security levels. A high-speed and lightweight implementation by employing a pipelined architecture has been presented for the AES-128 in [15]. In study [16] four different lightweight block ciphers (KLEIN, LED, Lilliput, and Ktantan) have been implemented with fullwidth and serial hardware. Several FPGA-based implementations of the PRESENT algorithm have been done in [17]–[20]. Two lightweight block ciphers have been implemented on Spartan3 FPGA in [17]. 16-bit data-path has been employed for the PRESENT algorithm and the round function is implemented using dual-port distributed random access memory (DRAM). Tay *et al.* [18] are used an 8-bit data path to reduce hardware size. Also, the substitution box (S-box) is implemented using a Boolean structure. Pandey *et al.* [19] are presented two architectures with 80-bit and 128-bit keys. The S-boxes of these architectures are implemented using a combinational logic circuit. They achieved a latency of 33 clock cycles. In reference [20] two 16-bit architectures for the present cipher are designed. However, memory addressing operations have been used to perform the permutation of this architecture. The HIGHT algorithm has been implemented with scalar and pipelined architectures on FPGA in [21]. A serialized and round-based design for the LED [22] and PHOTON is presented in [23], [24]. The hardware implementation of the RECTANGLE algorithm in [25] includes different datapaths with different data bus sizes. The lightweight RoadRunner has been implemented in [26] with 8-bit, 16-bit, and 32-bit datapath size on different platforms. The block cipher ANU has been implemented by [27], [28]. In [27] an 8-bit datapath is proposed and implemented. Dahiphale *et al.* [28] are implemented lightweight ANU block cipher at 4-bit, 8-bit, 16-bit, and 32-bit datapath sizes on different platforms. parallel substitution boxes (S-boxes) have been used by [28]. three different hardware architectures for the lightweight LiCi algorithm are presented by [29]. These architectures include serialized, a reduced datapath, and a pipelined architectures. Two architectures namely Khudra-I and Khudra-II for the lightweight block cipher Khudra are proposed in [30]. Finally, two FPGA designs which are parallel and 8-bit datapath are proposed in [13] for the KLEIN Block Cipher. An FPGA architecture and high execution-speed hardware IP-Core for the PRINCE algorithm, that produce the encrypted plaintext in one clock cycle, is designed in [31], [32]. Block ciphers have two structures that is substitution-permutation network (SPN) and Feistel network (FN). Table 1 presented related work of the lightweight algorithms with some its characteristics.

The motivation of this research is the increase in demand for limited resource devices such as smart cards and radio frequency identification (RFID) devices. These devices do not have any software programmable processors [27]. They are used in many applications that exchange sensitive data which may cause large harm if it were breached. Therefore avoiding incidents is important for these devices. Moreover, conventional cryptographic algorithms are not appropriate for such applications, especially the highly constrained environments. Consequently, the hardware implementation of lightweight ciphers is important in providing security for these devices. Moreover, these devices might require the manipulation of large data in real-time. Therefore, the trade-off between area and performance needs to be considered. So, it is important in any lightweight implementation to reduce the cost of hardware resources and increase performance.

Table 1. Lightweight algorithms with some its characteristics.

Algorithm	Ref.	Year	Structure	Advantages, Disadvantages
LED	[9]	2019	SPN	The authors have focused on the 64-bit key only not the 128-bit key.
PRINCE	[10]	2020	SPN	The author improved the area and the speed in this work.
PRINCE	[11]	2021	SPN	The author used the PRINCE algorithm in the bbh4 application for increasing security. However, the slice number is high.
SIMON	[12]	2019	FN	The authors focused on energy and power for the scalar and pipelined implementations.
KLEIN	[13]	2019	FN	The authors of proposed parallel and iterative architectures. However, A strong focus is placed on high-throughput implementations.
AES 32-bits	[14]	2017	SPN	The total number of cycles to encrypt a block in AES 32-bits architecture is 44 cycles. The authors saved the area by organizing the datapath.
AES-128	[15]	2021	SPN	The authors implemented AES-128 with high throughput pipelined architecture suitable for 5G communications.
PRESENT	[18]	2015	SPN	8-bit data-path to reduce hardware size with increase latency. The design has a latency of 295 cycles in total.
PRESENT	[19]	2017	SPN	The S-boxes of these architectures are implemented using a combinational logic circuit. They achieved a latency of 33 clock cycles
PRESENT	[20]	2016	SPN	Memory addressing operations have been used to perform the permutation of this architecture. The main goal is to obtain a design with a small area implementation. (latency 33,250,132)
HIGHT	[21]	2016	GFN	Two architectures have been designed with multiple rounds. These architectures include pipeline and scalar designs. However, the maximum frequency for these two architectures decreases when the number of rounds increases.
LED	[24]	2014	SPN	Three different architecture is proposed. However, the latencies of these architectures range from 256 to 1680 clock cycles for different platforms.
PHOTON	[24]	2014	Hash function	The author proposed three architectures (round-based, fully Serialized based on a single cell during each clock cycle, also fully serial, but based on the SRL16s and aiming at the smallest area possible
RECTANGLE	[25]	2019	SPN	The authors proposed novel architectures for the RECTANGLE block cipher. The latencies of these Different architectures are 26,101,126,151 and 251.
RoadRunner	[26]	2019	FN	The author implemented the RoadRunner cipher with three different datapath sizes. These architectures with latencies 13, 26 and 49. The authors balance the performance of these architectures.
ANU	[28]	2020	FN	The authors designed different datapaths for the ANU cipher. They used parallel substitution boxes(s-boxes) for high throughput and optimized area.
Khudra	[30]	2014	FN	Two variants of Khudra cipher with latencies 36 and 54
PRINCE	[31]	2014	SPN	The author implemented the PRINCE algorithm using FPGA with low area and high speed.
PRINCE	[32]	2014	SPN	The author designed IP-Core for the PRINCE algorithm using micro blaze processor.

In this paper, we have proposed and performed an efficient hardware implementation of 64-bit datapath architecture of lightweight block cipher ANU-II on the FPGA platform. Four different FPGA platforms, which include Spartan-3, Spartan-6, Virtex-4, Virtex-6 based on both Look-Up Table-4 (LUT-4) and Look-Up Table-6 (LUT-6) technologies have been used for hardware implementation of this cipher. Performance metrics of this activity have been compared with different architectures across all platforms. Although the number of rounds for the ANU-II cipher is 25, only 13 clock cycles are required to produce a 64-bit of block cipher. The main contribution in this paper is a novel architecture for the ANU-II cipher.

This paper is categorized: section 2.1 presents specifications and elaborates on all the components of the ANU-II block cipher with its key scheduling. Section 2.2 elaborates on the design of hardware architectures in this paper. Section 3 presents the obtained implementation results and provides a comparison with other related works. Also, section 3.2 represents power consumption in tabular and graphical format. Finally, section 4 outlines the conclusion of this paper.

2. RESEARCH METHOD

2.1. ANU-II algorithm

The flow of the design is crucial for the encryption algorithm. Since ANU–II is a Feistel structure, therefore, the input of 64-bit plaintext is divided into two 32-bit sub-blocks which are P_MSB and P_LSB. The most significant 32 bits are assigned to P_MSB whereas the least significant 32-bit are assigned to P_LSB. From each round of key scheduling 64 bits are extracted and divided into two 32 blocks as RKi1 (most significant 32 Bits of 128-bits Key for ith Round) and RKi2 (63 to 32 Bits of 128-bits Key for ith Round) which are XOR-ed at specific positions that appeared in the design in Figure 1 [8].

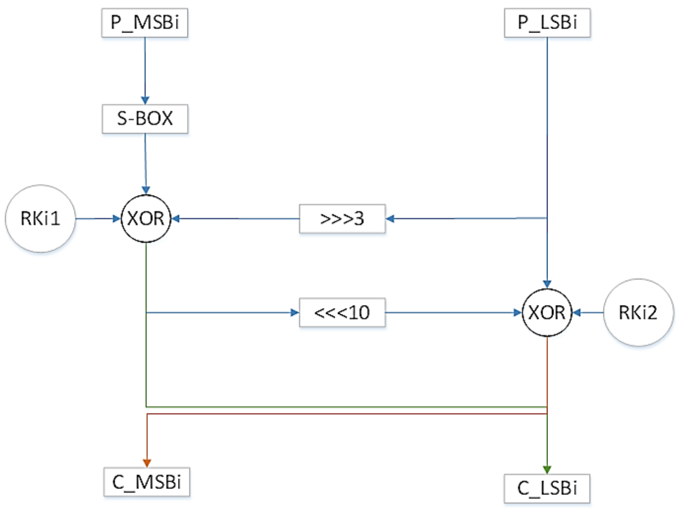


Figure 1. ANU-II cipher [8]

Where the symbols >>> and <<< denote circular right shift and circular left shift and the symbol $\oplus$ refers to an Exclusive OR operation.

2.1.1. The ANU-II layers

ANU-II algorithm is a 64-bit Feistel structure. That is the plaintext consists of 64 bit which is divided into two parts the most significant bit (MSB) part (32 bit) contain bits from (63 to 32) of plaintext and the least significant bit (LSB) part (32 bit) which contain bits from (31 to 0)of plaintext. The ANU-II algorithm support two keys which are 128 bit and 80 bit. The cipher has 25 rounds. Each round consists of one 4-4 SBOX, one circular right shift by 3 bit, one circular left shift by 10, and two XOR operations.

ANU cipher design has been modified by a group of researchers which results in an efficient version of the ANU cipher design called the ANU-II algorithm. An effort has been made to get efficient design metrics for limited resources devices in IoT and RFID tags [8]. The ANU-II cipher consists of the following components:

a. S-Box

The first component is the S-Box which is the only none linear layer in the ANU-II cipher. ANU-II uses 4-bit input to 4-bit output mapping S-Box. Table 2 shows the action of the S-Box used in the ANU-II cipher.

Table 2. S-BOX for ANU-II

X	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S(x)	E	4	B	1	7	9	C	A	D	2	0	F	8	5	3	6

b. Circular right shift by 3

The circular right shift implemented using wires only as showed in Figure 2. Where bits from 3 to 31 shifted to the right to constitute bits from 0 to 28. Also, bits from 0 to 1 rotated left to constitute bits from 29 to 31.

c. Circular left shift by 10

The circular left shift is implemented using wires only as showed in Figure 3. Where bits from 0 to 21 shifted to the left to constitute bits from 10 to 31. Also, bits from 22 to 31 are rotated to constitute bits from 0 to 9.

d. XOR operation

Since the plaintext is divided into two parts, and each part consists of 32 bits. The process of XOR on the left of the ANU-II cipher as shown in Figure 1 is among MSB 32 bits of plaintext and shifted to right LSB 32 bit of plaintext and a 32-bit result of the key schedule. In addition, the process of XOR on the right of the cipher in Figure 1 is among the result of shifting to the left previous XOR result LSB 32 bit of plaintext and the other 32-bit result of the key schedule.

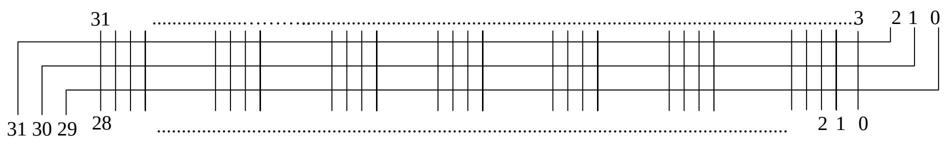


Figure 2. Circular shift right by 3

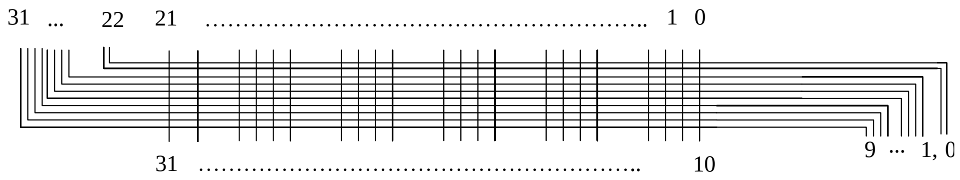


Figure 3. Circular left shift

2.1.2. Key schedule

As we said previously ANU-II support two keys of length 80 bit and 128-bit. In this implementation, we have used a 128-bit key length. The KEY schedule consists of the following components as shown in Figure 4 [8].

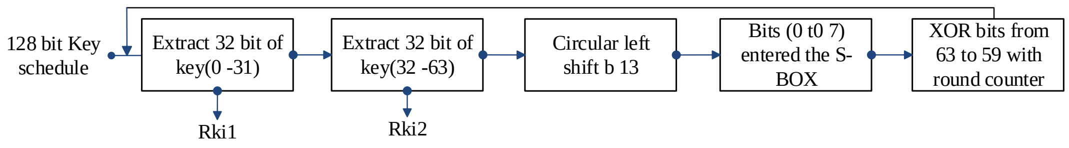

128 bit Key schedule
Extract 32 bit of key(0 -31)
Extract 32 bit of key(32 -63)
Circular left shift b 13
Bits (0 t0 7) entered the S-BOX
XOR bits from 63 to 59 with round counter
Rki1
Rki2

Figure 4. Key schedule structure

a. Circular left shift by 13: The circular left shift was implemented using wires only as we did in the encryption process for the left and right circular shifts but instead of the circular left shift by 10, we used circular left shift by 13 in the key schedule.
b. S-box: The same S-box used in encryption also it was used in the key schedule but only the bits from 0 to 7 will enter the S-box as shown in Figure 5.
c. XOR operation: Executed between the round counter and the bits of the key from 59 to 63.

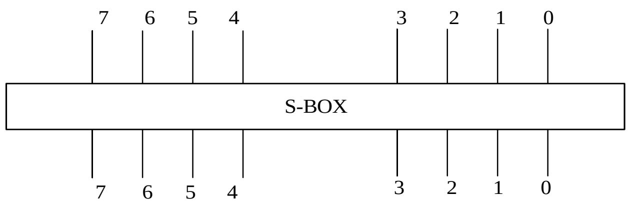


Figure 5. S-BOX in the KEY schedule

2.2. FPGA implementation of ANU-II

In our design, the main target is a small footprint area, low-cost hardware implementation, and high throughput. ANU-II consists of several components such as an S-box layer, 64 XOR, circular right or left shifts. S-BOX layer is the most expensive operation therefore it has been built using look-up tables (LUTs).

The designed architecture uses 8 S-boxes for data layer implementation. Also, two separate s-boxes are used for key scheduling. High throughput is achieved by this architecture compared to other round-based architectures of the standard ciphers as shown in Table 3. Only one clock cycle is required to perform two rounds of the algorithm. Since the ANU-II is 25 rounds, therefore, only 13 clock cycles are required to encrypt a block of 64-bit data.

Three state machines are required to implement the ANU-II algorithm as shown in Figure 6. In the first state (S0), the scheme loads the data block of 64-bit length and the key of 128-bit length. When the Reset signal is equal to one, the structure will reset itself and stays in state zero (S0). When the Reset signal is equal to zero and the clock (CLK) signal is equal to one, it will move execution to the second state (S1). In this state, the encryption operation and the key scheduling operation are performed. The round counter (RC) is incremented by one during each round. And, the execution remains in this state until the round counter (RC) is equal to 25. The execution moves to the third state (S2) when the RC is equal to 25. In this state, the cipher-text is generated. Also, the round counter is reinitialized to zero.

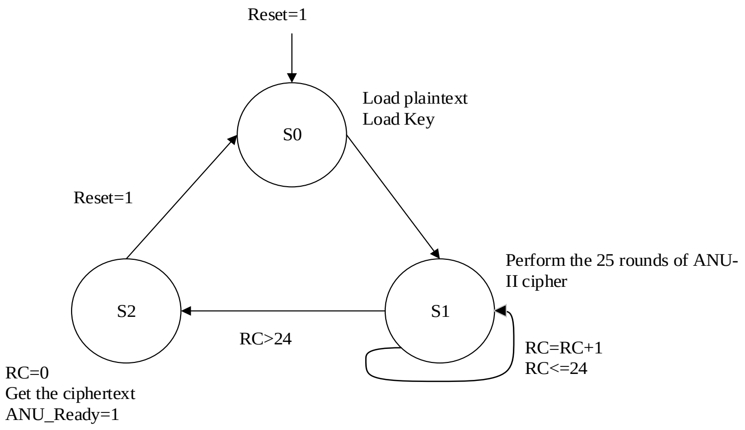
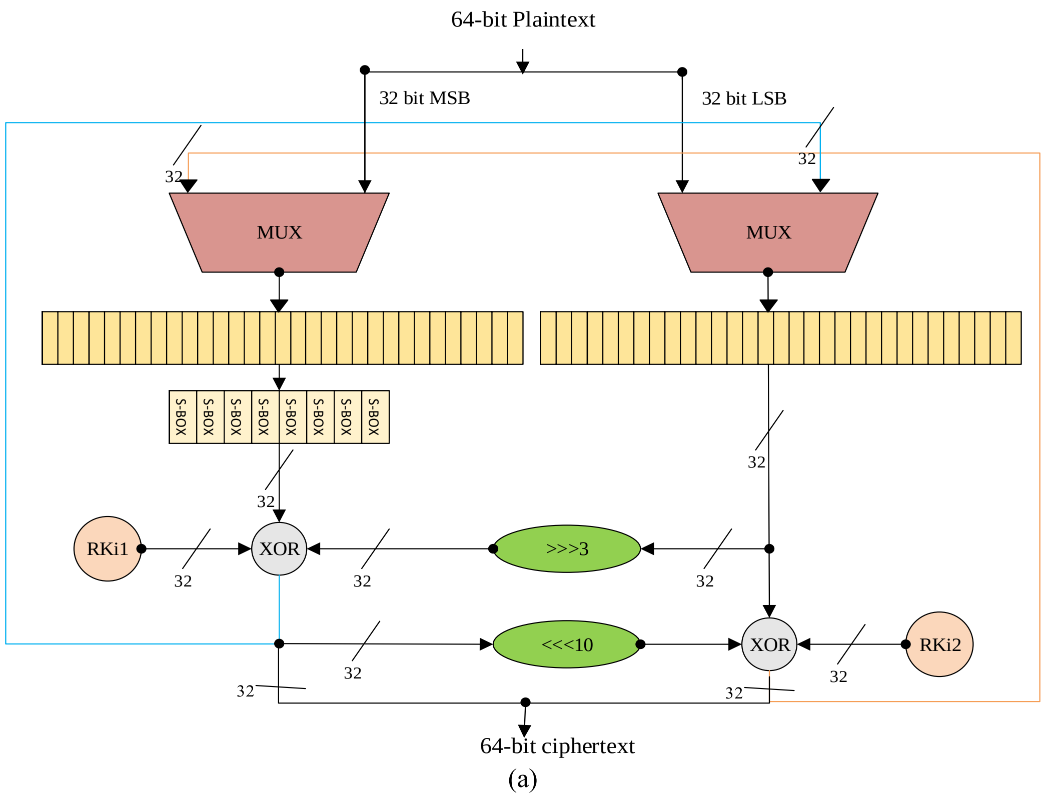


Figure 6. Three-state machine for ANU-II cipher

ANU-II is a modified version of ANU.ANU has been implemented using FPGA but ANU-II has not been implemented on FPGA until now as far as our deep research. Architecture has been designed for the ANU-II as shown in Figure 7(a). Firstly, the 64-bit plaintext is input to two multiplexers. The left multiplexer consumes 32 most significant bits (63:32). The right multiplexer consumes the least significant 32 bits (31:0). Secondly, the output of these multiplexers is consumed by two registers each of which contains 32-bits. These registers are necessary for storing intermediate results for other processing. Thirdly, the most significant 32-bits goes through eight S-Boxes. Each S-box contains four inputs (4 bits) and produces four outputs (4 bits). Fourthly, the result of these S-Boxes is XORed with the value RKi1 (the subkey generated from the key scheduling process) and the result of the right circular shift by 3 for the least significant 32 bits. The result of this XOR operation goes through a circular left shift by 10. Fifthly, the left significant 32-bits goes through other XOR operations with the RKi2(the subkey generated from the key scheduling process) and the result of circular left shift by 10. The results of these two XOR operations are finally swapped with each other. After, twenty-five round the ciphertext will be available on the output. Key scheduling is presented in Figure 7(b). Firstly, the 128-bit key entered a multiplexer. The output of this multiplexer is stored in an intermediate 128-bit register. This register is necessary for storing intermediate results. Secondly, the 32-bits from 31 down to 0 constitute RKi1 and from 63 down to 32 constitute RKi2. The steps of key scheduling operation firstly generate the two subkeys. Secondly, the 128-bit is going through a circular left shift process by 13. After that bit from 0 to 7 constitute the inputs for two S-boxes. Finally, the bits from 59 to 63 are XORed with a 5-bits round counter (RC) i.e. RC=0, 1…24.

In Figure 8 the block cipher interface of our design has been shown. The design involves eight ports; three ports for input the 64-bit plaintext which is divided as least significant 32 Bits of Plaintext for ith Round (P_LSBi) 32 bit and most significant 32 Bits of Plaintext for ith Round (P_MSBi) 32 bit and the 128-bit key. Also three ports for the clock signal and the control signal and the reset signal. Finally two ports for output 64-bit ciphertext which is also divided as least significant 32 Bits of ciphertext for ith Round (C_LSBi) 32 bit and most significant 32 Bits of ciphertext for ith Round (C_MSBi) 32bit. The click through rate (CTR) signal is used as enable signal for the architecture. The readability-strength-tone (RST) signal is necessary to reset the architecture.



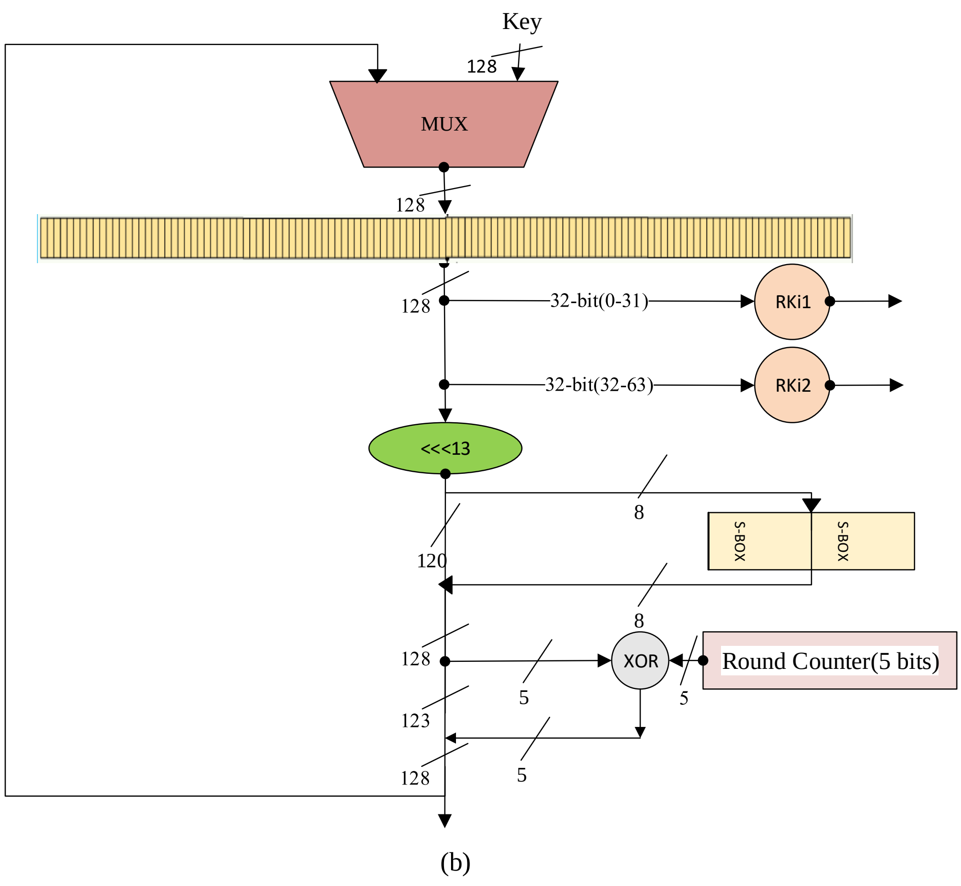


Figure 7. The hardware architecture: (a) the ANU-II encryption unit and (b) the key schedule

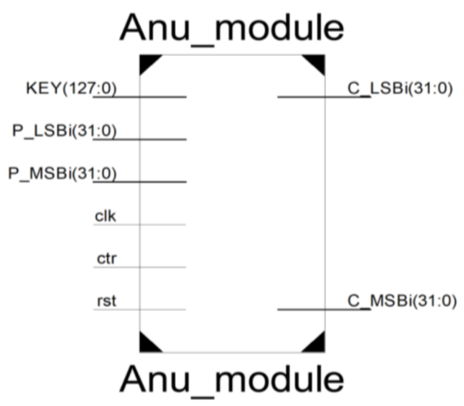


Figure 8. Top-level interface

In the Feistel structure, encryption and decryption are the same but reversal of the key scheduling is required in some casesz [5]. The decryption unit of ANU-II is similar to the encryption unit but the first Round would require Kn, the second round would require Kn-1, and so on. Where Kn, Kn-1...K0 are the subkeys result from the key schedule in decryption operation. Decryption data flow is shown in Figure 9. The input to decryption operation is the ciphertext and the key result from the last round of encryption operation. Only, 60 slices are required to implement the decryption process. Also, as an encryption process, the decryption process requires only 13 clock cycles to produce 64-bit of plaintext. The hardware components of the decryption unit consist of multiplexers, registers, inverse S-Boxes, XOR and shift operations. The hardware implementation of multiplexers, registers and XOR operation is the same as for the encryption unit. The hardware implementation of the shift operation is identical for the encryption unit but instead of a circular left shift, it will become a circular right shift and vice versa. Also, only wires are used to implement the shift operation in the decryption unit, to reduce hardware footprint. Look-up tables have been used to implement the inverse S-Box.

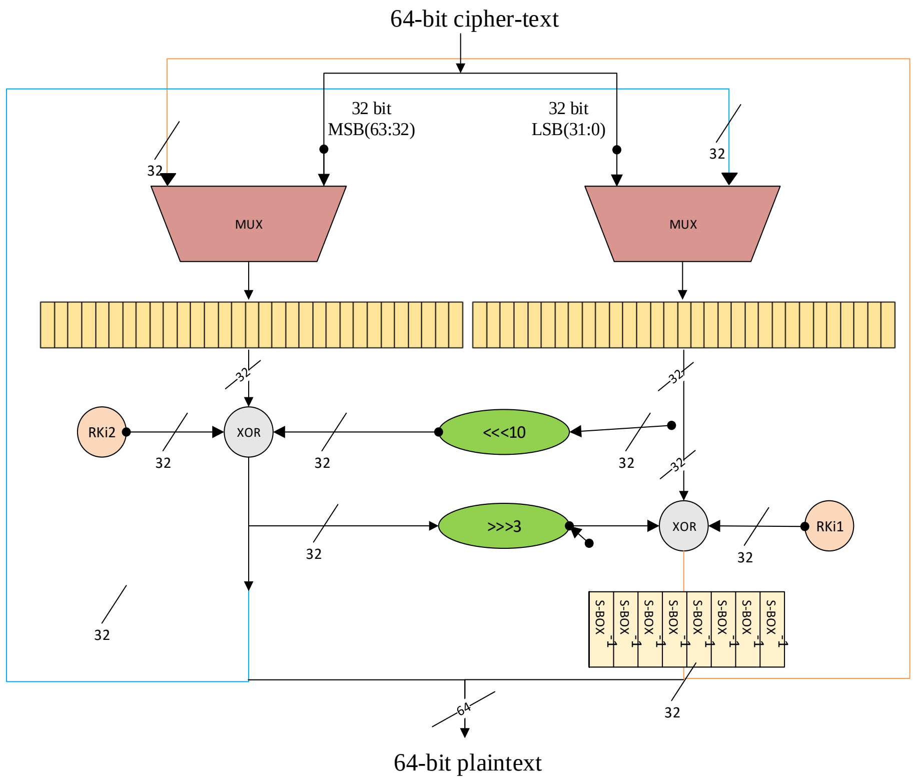


Figure 9. Decryption architecture

3. RESULTS AND DISCUSSION

3.1. Results and evaluation

The proposed architecture has been evaluated on four different FPGA platforms with different metrics like area, throughput, power consumption, energy, and latency. Table 3 shows the performance of the proposed architecture and also compares it with the implementation of standard lightweight ciphers. From Table 3 and previous studies, it can be showed that all the performance metrics highly depend on the platform used for the implementation [26]. Several existed FPGA implementation has been compared with hardware design proposed like ANU [27], ANU [28], LED [24], PRESENT(C1) [33], Piccolo [34], PRINCE [10], [11] algorithms. High throughput and good efficiency have been shown by our results with normal frequency speed as depicted in Table 3 as compared to other existed block ciphers in this table.

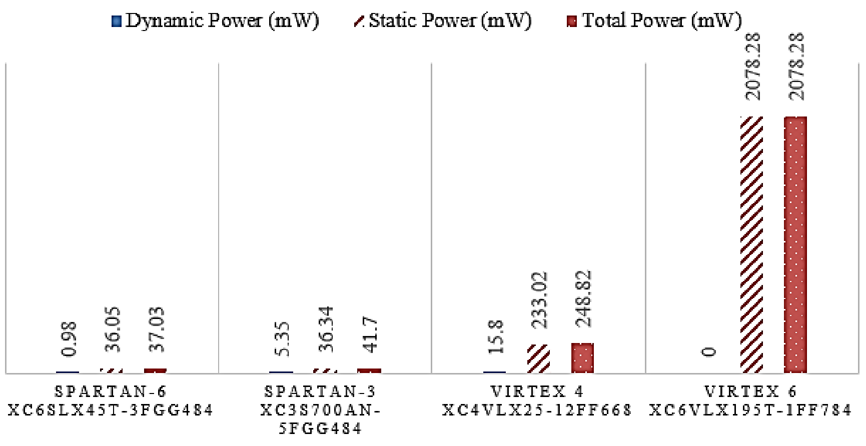
The design efficiency is measured as the ratio of the achieved throughput to the occupied slices. The throughput and the efficiency of the proposed work in this paper are higher than all other proposed works existed in Table 3. So, the efficiency is higher than its counterpart with at least twice for example the efficiency of the proposed Spartan-6 design is at least twice higher than the other two designs.

Table 3. Comparison between the ANU-II architecture and the round-based implementation of standard cipher

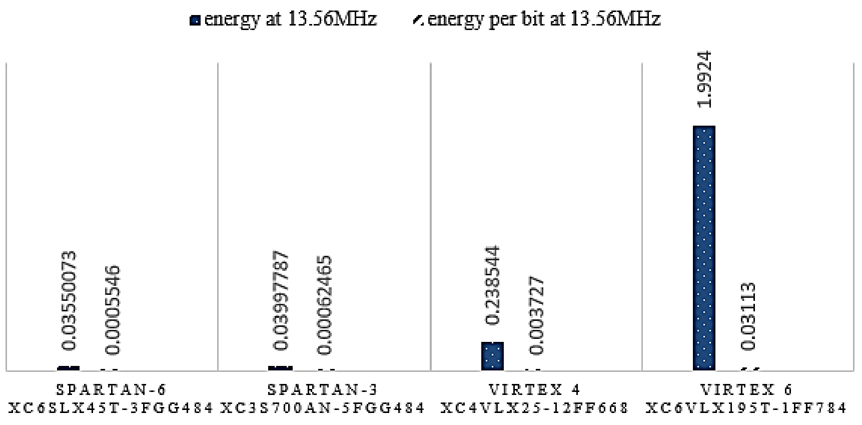
Device	Cipher/ Architecture	Data size (bit)	Key size (bit)	Flip Flops	LUT	Slices	Latency	Maximum frequency (MHz)	Throughput (Fmax)/ (Mbps)	Efficiency Mbps /slice
Spartan-3 xc3s700an-5fgg484	Proposed ANU-II	64	128	199	413	214	13	305.157	1502.31	7.0201
Spartan-3 xc3s700an-5fgg484	ANU [13]	64	128	210	460	265	125	243.253	124.54	0.46996
Spartan-3 (xc3s700an-5fgg484)	ANU(D4) [14]	64	128	199	513	272	25	262.123	671.03	2.467
Spartan-3 XC3S50pq208-5	Piccolo [34]	64	128	207	757	397	31	81.82	168.9	0.495
Spartan-3 (xc3s50-5)	LED(x) [24]	64	128	76	391	199	48	78.79	104.8	0.53
Spartan-3 (xc3s50-5)	LED(x)^2 [24]	64	128	76	444	227	48	87.63	116.54	0.51
Spartan-3 (xc3s50-5)	LED(x)^4 [24]	64	128	76	456	233	48	98.7	131.2	0.56
Spartan-3 (xc3s200-5ft256)	PRESENT(C1) [15]	64	128	200	381	191	55	179.950	209.4	1.096
Spartan-6 xc6slx45t-3fgg484	Proposed ANU-II	64	128	201	230	61	13	396.471	1951.86	31.997
Spartan-6 xc6slx45t-3fgg484	ANU [13]	64	128	216	357	93	125	266.766	136.58	1.4686
Spartan-6 (xc6slx45t-3fgg484)	ANU(D4) [14]	64	128	208	233	62	25	251.276	899.26	14.7419
Virtex 4 xc4vlx25-12ff668	Proposed ANU-II	64	128	200	493	253	13	547.72	2696.47	10.657
Virtex 4	PRINCE [10]	64	128	-	-	387	11	357.782	2081.639	5.378912
Virtex 4 FF668	PRINCE [11]	64	128	-	-	1026	1	31.715	2029	1.9
Virtex 6 xc6vlx195t-1ff784	Proposed ANU-II	64	126	199	229	83	13	778.21	3831.19	46.72
Virtex 6	PRINCE [10]	64	128	-	-	256	11	465.116	2706.13	10.571

3.2. Power consumption

Xpower analyzer is used to calculate both the energy and power consumption as shown in Figure 10. the static power and dynamic power for the four platforms used to implement the proposed architecture is shown in Figure 10(a). In addition, energy dissipation has been computed at the ISO frequency of RFID tags and smart cards (13.56 MHz) for each platform as shown in Figure 10(b).



(a)



(b)

Figure 10. Xpower analyzer is used to calculate (a) power consumption and (b) energy dissipation

4. CONCLUSION

The proposed work presents an efficient lightweight implementation for the lightweight block cipher ANU-II on different FPGA platforms using Xilinx ISE 14.7. This lightweight implementation is performed on four different FPGA platforms that include Spartan-3, Spartan-6, Virtex-4 and Virtex-6 based on both LUT-4 and LUT-6 technologies. An iterative looping architecture is employed for both the encryption and decryption tasks. Although the number of rounds for the ANU-II algorithm is 25 rounds, the clock cycles that need to produce the 64-bit ciphertext is only 13. The post-synthesis and implementation results were reasonable as compared with other studies. The proposed lightweight architecture produced high throughput and good efficiency. The power consumption is computed using XPower Analyzer. In terms of area and power consumption spartan 6 have the most compact area and lesser power consumption than the other three platforms. The future work is to create more architectures types that can be designed such as pipelined architecture to increase throughput or fully unrolled architecture to decrease latency.

REFERENCES

[1] K. A. McKay, L. Feldman, and G. A. Witte, “Toward standardizing lightweight cryptography,” *ITL Bulletin*, no. 1–4, 2017.
[2] Y. Benslimane and K. BenAhmed, “Efficient end-to-end secure key management protocol for internet of things,” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 7, no. 6, pp. 3622–3631, Dec. 2017, doi: 10.11591/ijece.v7i6.pp3622-3631.
[3] D. Ciuonzo, G. Gelli, A. Pescape, and F. Verde, “Decision fusion rules in ambient backscatter wireless sensor networks,” in *2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Sep. 2019, pp. 1–6, doi: 10.1109/PIMRC.2019.8904358.
[4] D. Ciuonzo and P. Salvo Rossi, “DECHADE: detecting slight changes with hard decisions in wireless sensor networks,” *International Journal of General Systems*, vol. 47, no. 5, pp. 535–548, Jul. 2018, doi: 10.1080/03081079.2018.1455192.
[5] A. H. A. Al-ahdal and N. Deshmukh, “A systematic technical survey of lightweight cryptography on IoT environment,” *International Journal of Scientific & Technology Research*, vol. 9, no. 3, pp. 6246–6261, 2020.
[6] D. Ciuonzo, P. S. Rossi, and P. K. Varshney, “Distributed detection in wireless sensor networks under multiplicative fading via generalized score tests,” *IEEE Internet of Things Journal*, vol. 8, no. 11, pp. 9059–9071, Jun. 2021, doi:

10.1109/JIOT.2021.3056325.

[7] H. Darvishi, D. Ciuonzo, E. R. Eide, and P. S. Rossi, “Sensor-fault detection, isolation and accommodation for digital twins via modular data-driven architecture,” *IEEE Sensors Journal*, vol. 21, no. 4, pp. 4827–4838, Feb. 2021, doi: 10.1109/JSEN.2020.3029459.

[8] V. Dahiphale, G. Bansod, and J. Patil, “ANU-II: A fast and efficient lightweight encryption design for security in IoT,” in *2017 International Conference on Big Data, IoT and Data Science (BID)*, Dec. 2017, pp. 130–137, doi: 10.1109/BID.2017.8336586.

[9] M. Al-Shatari, F. A. Hussin, A. A. Aziz, G. Witjaksono, M. S. Rohmad, and X.-T. Tran, “An efficient implementation of LED block cipher on FPGA,” in *2019 First International Conference of Intelligent Computing and Engineering (ICOICE)*, Dec. 2019, pp. 1–5, doi: 10.1109/ICOICE48418.2019.9035193.

[10] B. Rashidi, “Low-cost and two-cycle hardware structures of PRINCE lightweight block cipher,” *International Journal of Circuit Theory and Applications*, vol. 48, no. 8, pp. 1227–1243, Aug. 2020, doi: 10.1002/cta.2832.

[11] 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, Mar. 2021, doi: 10.1088/1742-6596/1818/1/012216.

[12] S. Abed, R. Jaffal, B. Mohd, and M. Alshayeji, “FPGA modeling and optimization of a SIMON lightweight block cipher,” *Sensors*, vol. 19, no. 4, Feb. 2019, doi: 10.3390/s19040913.

[13] P. Singh, B. Acharya, and R. K. Chaurasiya, “High throughput architecture for KLEIN block cipher in FPGA,” in *2019 9th Annual Information Technology, Electromechanical Engineering and Microelectronics Conference (IEMECON)*, Mar. 2019, pp. 64–69, doi: 10.1109/IEMECONX.2019.8877021.

[14] D.-H. Bui, D. Puschini, S. Bacles-Min, E. Beigne, and X.-T. Tran, “AES datapath optimization strategies for low-power low-energy multisecurity-level internet-of-things applications,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 12, pp. 3281–3290, Dec. 2017, doi: 10.1109/TVLSI.2017.2716386.

[15] P. Visconti, R. Velazquez, S. Capoccia, and R. De Fazio, “High-performance AES-128 algorithm implementation by FPGA-based SoC for 5G communications,” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 5, pp. 4221–4232, Oct. 2021, doi: 10.11591/ijece.v11i5.pp4221-4232.

[16] C. Marchand, L. Bossuet, and K. Gaj, “Ultra-lightweight implementation in area of block ciphers,” in *Foundations of Hardware IP Protection*, L. Bossuet and L. Torres, Eds. Cham: Springer International Publishing, 2017, pp. 177–203.

[17] P. Yalla and J.-P. Kaps, “Lightweight cryptography for FPGAs,” in *2009 International Conference on Reconfigurable Computing and FPGAs*, Dec. 2009, pp. 225–230, doi: 10.1109/ReConFig.2009.54.

[18] J. J. Tay, M. L. D. Wong, M. M. Wong, C. Zhang, and I. Hijazin, “Compact FPGA implementation of PRESENT with Boolean S-Box,” in *2015 6th Asia Symposium on Quality Electronic Design (ASQED)*, Aug. 2015, pp. 144–148, doi: 10.1109/ACQED.2015.7274024.

[19] J. G. Pandey, T. Goel, and A. Karmakar, “An efficient VLSI architecture for PRESENT block cipher and its FPGA Implementation,” 2017, pp. 270–278.

[20] C. A. Lara-Nino, M. Morales-Sandoval, and A. Diaz-Perez, “Novel FPGA-based low-cost hardware architecture for the PRESENT block cipher,” in *2016 Euromicro Conference on Digital System Design (DSD)*, Aug. 2016, pp. 646–650, doi: 10.1109/DSD.2016.46.

[21] B. J. Mohd, T. Hayajneh, Z. A. Khalaf, and K. M. Ahmad Yousef, “Modeling and optimization of the lightweight HIGHT block cipher design with FPGA implementation,” *Security and Communication Networks*, vol. 9, no. 13, pp. 2200–2216, Sep. 2016, doi: 10.1002/sec.1479.

[22] J. Guo, T. Peyrin, A. Poschmann, and M. Robshaw, “The LED block cipher,” in *International Workshop on Cryptographic Hardware and Embedded Systems CHES 2011*, 2011, pp. 326–341.

[23] J. Guo, T. Peyrin, and A. Poschmann, “The PHOTON family of lightweight hash functions,” in *Annual Cryptology Conference*, 2011, pp. 222–239.

[24] N. Nalla Anandakumar, T. Peyrin, and A. Poschmann, “A very compact FPGA implementation of LED and PHOTON,” in *International Conference on Cryptology in India*, 2014, pp. 304–321.

[25] V. Dahiphale, H. Raut, and G. Bansod, “Design and implementation of novel datapath designs of lightweight cipher RECTANGLE for resource constrained environment,” *Multimedia Tools and Applications*, vol. 78, no. 16, pp. 23659–23688, Aug. 2019, doi: 10.1007/s11042-019-7587-3.

[26] P. Pachange and G. Bansod, “A fast and efficient datapath designs of lightweight cipher RoadRunner on FPGA’s for resource constrained environments,” in *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, Oct. 2019, pp. 65–72, doi: 10.1109/IOTSMS48152.2019.8939241.

[27] V. Dahiphale, G. Bansod, and A. Zambare, “Lightweight datapath implementation of ANU cipher for resource-constrained environments,” in *Intelligent Computing - Proceedings of the Computing Conference*, 2019, pp. 834–846.

[28] V. Dahiphale, G. Bansod, A. Zambare, and N. Pisharoty, “Design and implementation of various datapath architectures for the ANU lightweight cipher on an FPGA,” *Frontiers of Information Technology & Electronic Engineering*, vol. 21, no. 4, pp. 615–628, Apr. 2020, doi: 10.1631/FITEE.1800681.

[29] N. Ayesha and B. Acharya, “VLSI implementation of LiCi cipher,” *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, vol. 8, no. 6, pp. 1184–1191, 2019.

[30] S. Kolay and D. Mukhopadhyay, “Khudra: A new lightweight block cipher for FPGAs,” in *International Conference on Security, Privacy, and Applied Cryptography Engineering*, 2014, pp. 126–145.

[31] Y. A. Abbas, R. Jidin, N. Jamil, M. R. Z’aba, M. E. Rusli, and B. Tariq, “Implementation of PRINCE algorithm in FPGA,” in *Proceedings of the 6th International Conference on Information Technology and Multimedia*, Nov. 2014, pp. 1–4, doi: 10.1109/ICIMU.2014.7066593.

[32] Y. Amer Abbas, R. Jidin, N. Jamil, M. Reza Z\’aba, and M. Ezanee Rusli, “PRINCE IP-core on field programmable gate arrays (FPGA),” *Research Journal of Applied Sciences, Engineering and Technology*, vol. 10, no. 8, pp. 914–922, Jul. 2015, doi: 10.19026/rjaset.10.2447.

[33] N. Hanley and M. ONeill, “Hardware comparison of the ISO/IEC 29192-2 block ciphers,” in *2012 IEEE Computer Society Annual Symposium on VLSI*, Aug. 2012, vol. 8885, pp. 57–62, doi: 10.1109/ISVLSI.2012.25.

[34] A. Mhaouch, W. Elhamzi, and M. Atri, “Lightweight hardware architectures for the piccolo block cipher in FPGA,” in *2020 5th International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)*, Sep. 2020, pp. 1–4, doi: 10.1109/ATSIP49331.2020.9231586.

BIOGRAPHIES OF AUTHORS

Yousif Nihad Hatif received the B.Sc. (1st Class Hons.) degree in Computer Engineering from University of Diyala, Iraq, in 2012. He is a staff member in Diyala University, Iraq. He is currently a M.Sc. student in Al Iraqia University, College of Engineering, Iraq. His research interests are in computer engineering, Web applications, lightweight cryptography, FPGA, artificial intelligence and image processing. He can be contacted at email: yousifnh1989@gmail.com.

Yasir Amer Abbas received the B.Sc. (1st Class Hons.) and M.Sc. degrees in Computer Engineering from University of Technology, Iraq, in 2000 and 2005, respectively. His Ph.D. in computer engineering from Universiti Tenaga Nasional (UNITEN), Malaysia 2016. He is Assistance Professor at College of Engineering, University of Diyala. His research interests are in computer engineering, including, embedded systems hardware-software co-design and cryptography application, FPGA, computer architecture. He has authored and co-authored numerous publications in international journals and conferences. He is a senior member of the IEEE and the IEEE Computer Society since 2012. He can be contacted at email: yasiramerabbas@gmail.com.

Mudhafar Hussein Ali received his B.Sc. in electrical and electronics engineering and his M.Sc. in laser and optoelectronics engineering from AL Rasheed College of Engineering and Science, University of Technology, Iraq, in 1996 and 2004, respectively. His employment experience includes the Research and Development Center in Ministry of Science and Technology. His special fields of interest include fiber amplifiers, laser systems and applications. By 2004, he was a chief of engineers at the Ministry of Science and Technology/Renewable Energy Center. In 2015, he received his Ph.D. in communications engineering from Universiti Tenaga Nasional. Since 2016, he has been a staff member in the Network Engineering Department, College of Engineering, Al Iraqia University, Iraq. He can be contacted at email: muthafarh@yahoo.com.