Intelligent Automation & Soft Computing
DOI:10.32604/iasc.2022.020681
Article

Tech Science Press

Light-Weight Present Block Cipher Model for IoT Security on FPGA

R. Bharathi* **and N. Parvatham**

Department of Electronics & Communication Engineering, PRIST University, Tamilnadu, 613403, India
*Corresponding Author: R. Bharathi. Email: bharathir2474@gmail.com
Received: 03 June 2021; Accepted: 08 July 2021

Abstract: The Internet of Things (IoT) plays an essential role in connecting a small number of billion devices with people for diverse applications. The security and privacy with authentication are challenging work for IoT devices. A light-weight block cipher is designed and modeled with IoT security for real-time scenarios to overcome the above challenges. The light-weight PRESENT module with the integration of encryption (E)-decryption (D) is modeled and implemented on FPGA. The PRESENT module has 64-bit data input with 80/128/256-bit symmetric keys for IoT security. The PRESENT module performs16/32/64 round operations for state register and key updation. The design mainly uses Substitution-permutation (SP) network for state updation. The permutation layer is used to create more diffusion and confusion in the state for unauthorized access. The results and analysis of the PRESENT-80/128/256 are designed using Verilog-HDL with Xilinx Environment and implemented on Artix-7 FPGA. The PRESENT-80/128/256 module is compared with similar recent works with performance improvements. Similarly, the proposed work is compared with different light-weight algorithms with improvements for better security in IoT devices. The proposed PRESENT-256 module with 64-rounds on Artix-7 FPGA utilizes less than 2% Chip area (Slices and LUTs), works at 412.4 MHz frequency, and consumes 192 mW total power and 0.58 Mbps/slice of hardware efficiency.

Keywords: IoT devices; security; present block cipher; lightweight; encryption; decryption; FPGA

1 Introduction

The IoT is used to connect social networks, allowing devices and people to interact with information sharing. At present, around 8.4 billion devices are interconnected globally, and by 2022 it is expected to be around 20 billion. The usage of IoT applications is increasing day by day, and people are making use of them effectively worldwide. The IoT is one of the futures markets to expand the digital economy to the next level [1]. The IoT industry expects economic growth in terms of revenue from billions in 2018 to trillions by 2024. There is a considerable difference and tremendous changes in past, present, and future IoT architectures. The past IoT architecture contains many devices and is connected individually to the cloud system. The cloud system provides data processing, storage, and other interconnection services to the devices individually. The present IoT architecture has different device groups, and all are connected

to the cloud system through standard node devices. The same categories of devices are grouped and communicated with each other. In future IoT architectures, all the devices are connected to the internet and communicate with each other directly. Social IoT is introduced to connect the different social networking users to interconnect to things. So, with the help of the internet, the user can share their devices directly [2]. The IoT supports a wide range of intelligent and diverse applications like smart cities, innovative environments, security and emergencies, smart grids and smart metering, smart agriculture, smart retail, and many more.

The authentication with security, Interoperability, and privacy issues is a more challenging task in IoT devices. The future IoT applications can't perform their operations with high demand by not providing an interoperable, trusted IoT ecosystem. The IoT has to build better trust and create a suitable ecosystem. The internet, mobiles, and wireless sensor nodes (WSN) are common resources to create security issues generally. Apart from the security issues, The IoT faces other major challenging issues like authentication, information storage, management issues, privacy issues, and many more. There is always a significant challenge to provide security to IoT devices than the typical devices (information technology (IT) devices). Normal (IT) devices are generally resource-rich devices implemented with complex algorithms to provide security and other capabilities. The IT devices are used homogeneous technology for higher security. The IoT devices have device limitations based on software and hardware and only lightweight security algorithm based on the provision. The heterogeneous technology is incorporate with IoT for data security in an enormous amount [3].

In general, The IoT application has four layers, namely: Sensing layer, Network Layer, Middle-ware Layer, and Application Layer. Each layer is used for different IoT applications using different technology platforms, which causes enormous security threats and issues. The four Layers used in different IoT applications are represented in Fig. 1 with possible devices and technologies [4–6].

The sensing Layers mainly contain sensors and actuators. The sensors sense the physical process or changes in the environment and send the information to the other devices. At the same time, Actuators receives the sensing data and responsible for controlling and moving mechanism. Many sensors like ultrasonic sensors, temperature, pressure, heat, smoke detector, and humidity are available and are used to sense the physical environment. The Network Layer is used to transmit the data information received from the sensing layer to the control unit to process the operation. The Middleware layer is used to provide storing and computing features. This layer provides an abstract layer between network and application layers. The application layer's requirements are fulfilled by the middleware layer using the Application programming interface (API). Finally, the application layer provides required services to the end-users directly. Intelligent applications like homes, grids, cities, meters, healthcare, transport and buildings appear in this layer. These four layers are caused by possible threats and are tabulated in Tab. 1.

The security solutions for IoT using edge computing and machine learning are recent trends for low-end IoT devices. The IoT data is secured using scalable and end-end data collection modeling using intelligent data collection unit, actuation, and automation model, along with monitoring and analytics unit. Furthermore, the IoT devices are configured based on on-demand security with security module specification, interface specification, security service profile, device profile, device image generation unit with multiple devices [7]. The FPGA is also provided security to the trusted devices by configuring the bit-stream encryption and authentication. Furthermore, the FPGA provides features by using a unique identifier, Physically Unclonable Function (PUF), program and JTAG intercepts, key clear and device precise, Voltage and temperature monitors along with bit-stream scrubbing [8]. In addition, there are many security algorithms available for securing the IoT data includes Tiny XTEA [9], light encryption device (LED) [10], Hummingbird Algorithm (HB) [11], SPECK Block cipher [12], SIMON cipher [13], AES, DES, Triple DES [14], SIoT [15] and many more. However, in these, only lightweight ciphers are considered for IoT device security.

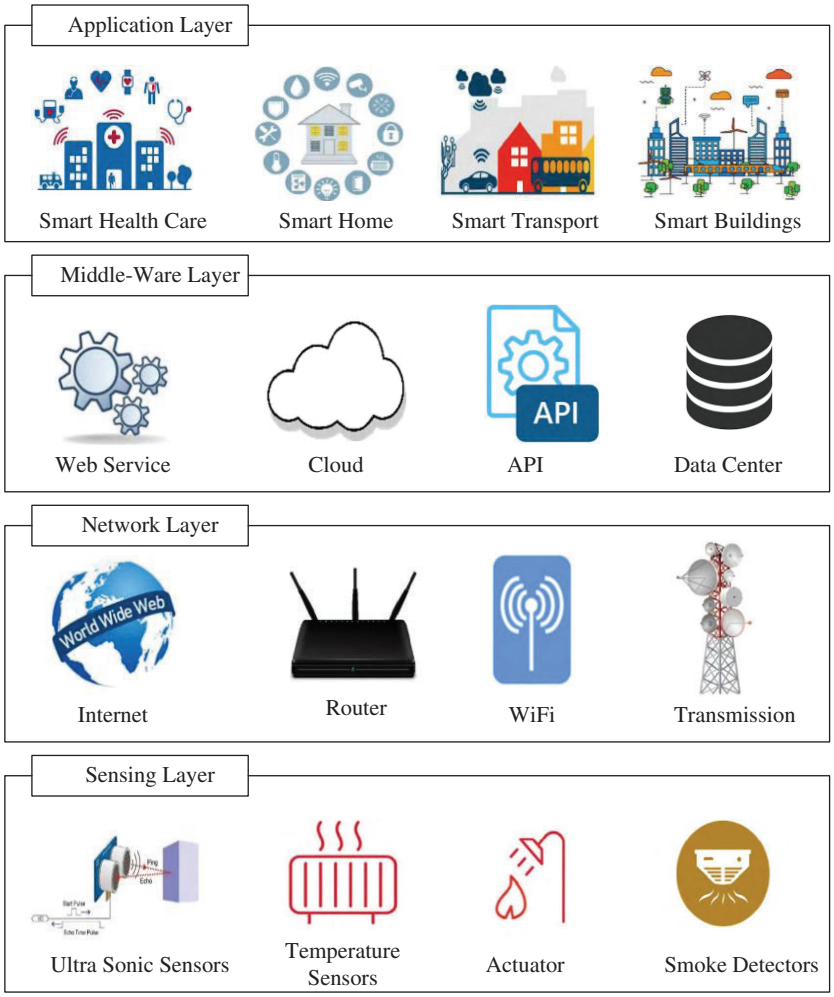

Figure 1: Layers used in IoT applications

Table 1: List of possible attacks in IoT layers

IoT layers	Possible attacks
Sensing layer	Side-channel attacks, eavesdropping booting vulnerabilities, false data injections, node capture attacks, sleep deprivation attacks, code injection attacks
Network layer	Common attacks, data transmit attacks, dos/ddos attacks, routing, access attacks, unlawful attacks, phishing attacks
Middle-ware layer	Men in-middle attack, signature attack, flooding attack (cloud), sql injection, cloud malware injection attacks
Application layer	Reprogram attacks, service interruption attacks, access control attacks, ddos attacks, code injection attacks, data theft attacks

Motivation: The security incorporation for IoT-based devices is essential to secure the received data information with privacy. Incorporating security and privacy features in IoT-based devices is challenging for several reasons [16]. Firstly, the IoT device's processor is limited and can't perform multiple tasks

within the limited timeline. Secondly, IoT devices are battery-operated, and it isn't easy to produce the low-power while incorporating security algorithms. Third, the security algorithm's implementation cost should be low to extend the other device's incorporation. Lastly, the advanced security algorithms usage in an IoT environment is suitable and affects the computational performance and increases the hardware complexity. The lightweight algorithms are introduced to resolve the above problems. The lightweight cipher algorithms provide better security with low cost and excellent system performance in a short duration. The lightweight block ciphers are suitable and provide security for low-end IoT devices in real-time applications without affecting unauthorized user's data.

Contributions: In this manuscript, an efficient PRESENT-80/128/256 lightweight cryptographic modules for encryption and decryption are presented with low Latency and High Throughput. The contribution of the proposed work is highlighted as follows:

- The existing PRESENT ciphers are working with 32 round operations with key. The proposed PRESENT cipher architecture is modified and configurable to work with 16/32/64 round operations individually with the key module.
- The proposed work introduces a 256-bit key in PRESENT cipher to strengthen the security features.
- The PRESENT-80/128/256 with different 16/32/64 rounds are works parallel with the key updation mechanism to improve the latency parameter and provide high throughput and hardware efficiency.
- The proposed PRESENT-80/128/256 work is synthesized individually and compares the performance parameters with existing PRESENT and lightweight ciphers with better improvements.

The manuscript is organized as follows: Section 2 presents the review of existing lightweight block ciphers, including recent PRESENT algorithms and their limitations. Section 3 describes the hardware architecture of PRESENT-80/128/256 encryption and decryption modules with proper key updation. Section 4 highlights the Results and comparative analysis of the proposed work. Finally, it concludes the overall work with improvements in Section 5 with future work.

2 Related Work

This section elaborates on the different cipher algorithms and recent PRESENT light-weight algorithms for different applications using different environments and highlights the limitations of these works. Mohd et al. [17] present the comparative discussion of the light-weight block ciphers for the lower-constrained devices. The list of light-weight algorithms with key size, block size, the number of rounds used, and the structure type is tabulated in detail. The implementation of ciphers with software and hardware platforms is discussed in detail with its performance metrics. The performance of hardware-based ciphers is compared with area, power, throughput, and energy factors. The open research issues like performance modeling, security metrics, software coding style, and hardware trojan are discussed. Edwar et al. [18] discuss the PRESENT algorithm and its performance evaluation on the FPGA platform. The hardware architecture of PRESENT with data path units is presented. It was designed PRESENT cipher (Both 80-bit and 128-bit Keys) using 64-bit-data size with 32-standard rounds. The proposed work is extended to 256-bit key with flexible 16/32/64-rounds with 64-bit-data size to improve the security aspects. The 256-bit key with 64-bit rounds in PRESENT cipher is complex combination for unauthorized users.

The work evaluates and compares the performance parameters like Latency, Throughput, CMOS process, and Area with existing approaches. Azari et al. [19] explain the PRESENT cipher model on the FPGA platform using 80-bit and 128-bit key mechanisms. The mathematical expression of the block cipher is analyzed in detail with architecture. The simulation waveforms of the 80-bit and 128-bit key mechanisms are discussed, along with resource comparison.

Thorat et al. [20] present the new hybrid cryptosystem with light-weight features for the IoT infrastructure. The hybrid approach includes the s-box of the PRESENT is integrated with permutation instruction-based PREMS cipher. The work is analyzed both in software and hardware environment with the realization of the performance metrics. Chom Thungon et al. [21] discuss the comparative analysis of both the AES and PRESENT Block ciphers for personal area network-based IoT applications. The performance comparison of hardware characteristics with different devices and memory consumption are discussed. The work concludes that the PRESENT cipher gives better performance in memory consumption than the AES cipher. Aragona et al. [22] discuss the PRESENT and other light-weight ciphers (PRINT and RECTANGLE) in detail with mathematical proofs. Pei et al. [23] explain the light-weight block ciphers usage and its performance, security trade-off in the Industrial Wireless sensor network (WSN). The Industrial WSN for factory automation is discussed using light-weight block ciphers (KLEIN, L-block cipher, PRESENT, Piccolo, HIGHT). The memory usage and throughput factors are discussed for all the light-weight block ciphers in the Industrial WSN system.

Jangra et al. [24] explains the implementation of the PRESENT and CLEFIA light-weight block ciphers in python and analyzes the performance metrics. The metrics like throughput, resource utilization, and security strength are discussed for both the ciphers. The PRESENT cipher gives less memory utilization and less throughput than the CLEFIA cipher. Rashidi [25] discuss the light-weight ciphers like LED, SIMON, and PRESENT modules with flexible hardware architectures. The work results are implemented on the ASIC platform with 180 nm CMOS Technology and are compared with existing similar works with constraints improvements. Jenny et al. [26] present the compact S-Box of block ciphers like LED, GIFT, and PRESENT for low-constrained applications. The S-Box designs are optimized using Karnaugh mapping to reduce the gate count values for these three block ciphers. Jang et al. [27] discuss the symmetric key cryptographic algorithms like PRESENT and GIFT with quantum computations. The reversible logic gates-based architecture is constructed for both the PRESENT and GIFT ciphers. The quantum resources like the number of gates used and circuit paths are discussed and compared with similar works.

Most of the PRESENT algorithms are conventional with iterative approaches and RAM or LUT-based key scheduling from these review analyses. These approaches consume more Chip areas and affect the system performance, and not suitable for IoT devices.

3 Proposed Work

The light-weight PRESENT encryption and decryption modules for 80/128/256-bit keys are presented in this section. The proposed PRESENT block cipher is well suited for IoT security in real-time scenarios.

3.1 PRESENT-80/128/256 Encryption Module

The PRESENT-80/128/256 encryption module is represented in Fig. 2. It mainly includes 64-bit plain text, Substitution-permutation (SP) network with a 16/32/64-rounds state updation, along with an 80/128/256-bit key updation unit. The SP network mainly contains 16 Substitution Boxes (S_box_Func) and Permutation Layer (PLayer). Initially, the Key load (kld) is activated, the 80/128/256-bit key is stored in the key register. When the key load is zero and load (ld) is activated, the state transition starts, the input 64-bit Plain-text is stored temporarily in the state register for the initial round. Suppose the load signal becomes zero for the next 16/32/64-rounds of operations. The state register data is XOR with 64-bit key register data as a round key and generates the SP Network 64-bit input. The MSB bits of 64-bit is considered for round key input out of 80/128/256-bit key register output. The SP network of 64-bit input divides the 4-bit Data individually 16 times and generates 16 S-boxes. The main aim of this work is to design a Light weight PRESENT-80/128/256 cryptographic modules for encryption and decryption with low Latency and High Throughput. The lightweight cipher algorithms provide better security with low

cost and excellent system performance in a short duration. The PRESENT lightweight block ciphers are suitable and provide security for low-end IoT devices in real-time applications without affecting unauthorized user's data. The Contribution section provides the main objective of the proposed work.

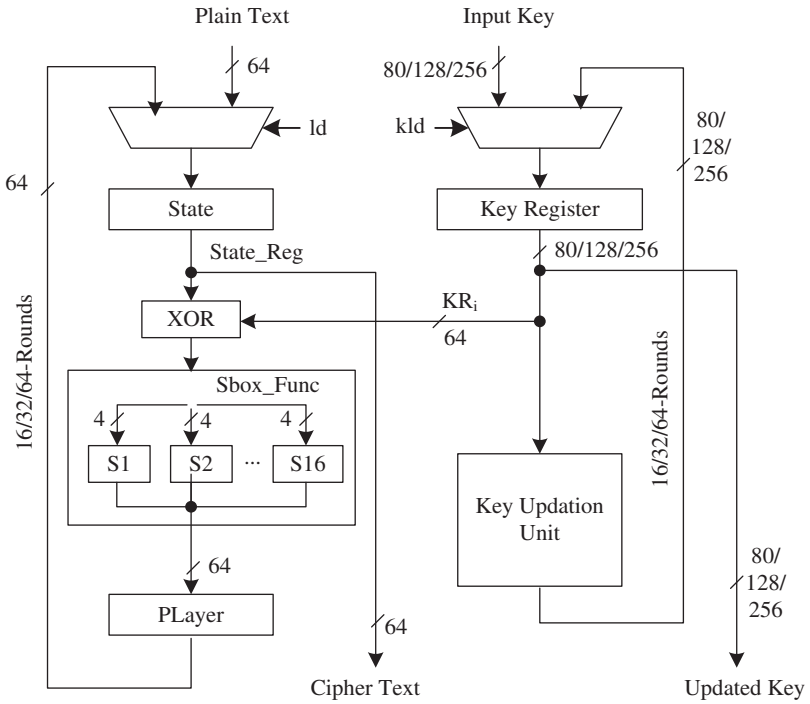

Figure 2: PRESENT-80/128/256 encryption module: pseudocode and hardware architecture

The key register (KR_i) provides the 64-bit key $KR_i = k_{63}^i \ldots k_0^i$ for $0 \leq i \leq 15/31/63$. The state register receives the KR_i with current State_Reg ($sr_{63} \ldots sr_0$) and performs the XOR operation using the below Eq. (1) till $0 \leq j \leq 63$.

$$sr_j = sr_j \oplus k_j^i \tag{1}$$

The State_Reg values are updated in Sbox_Func and perform the S-BOX operation of PRESENT-80/128/256, and it is tabulated in Tab. 2.

Table 2: PRESENT-80/128/256—Sbox_Func for encryption

In	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Out	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

```
generate Rounds_with_Keys ()
For (i = 0; i < 15/31/63; i = i+1;
    XOR (State_Reg, KRi)
    Sbox_Func (State_Reg)
        PLayer (State_Reg)
end for;
```

The SP inputs are replaced with S-box table data and generate 16 individual S-BOX outputs. Each S-BOX input and outputs are 4-bits. The Sbox_Func currently has 64-bit Data ($sr_{63}\ldots sr_0$) and decomposes it into sixteen 4-bit blocks ($b_{15}\ldots b_0$). Where $b_i = sr_{4*i+3} \parallel sr_{4*i+2} \parallel sr_{4*i+1} \parallel sr_{4*i}$ for $0 \leq i \leq 63$. The Sbox_Func [b_i] similarly updates the Sbox values and inputs them to the permutation layer. The permutation layer (PLayer) receives the 64-bit S-BOX outputs and creates more confusion and diffusion using the Permutation table [28]. The P-table data replaces the S-BOX data for the next 16/32/64-rounds and updates to the state register. Thus, after successive 16/32/64-rounds of clock cycles, the state register generates 64-bit ciphertext as PRESENT-80/128/256 cipher output. The Key updation unit updates the round key 16/32/64-times parallel, generates the updated 80/128/256-bit key data, and inputs it to the decryption module for reverse key updation. The detailed key updation for encryption and decryption for PRESENT-80/128/256-bit are described with hardware modeling in the next section.

In this work, the complete work is carried on FPGA with implementation. The Power reports are generated using Xilinx X-power analyzer. In FPGA, the total power is divided into two parts: Quiescent (Static) and dynamic power. The Quiescent (Static) power utilization is obtained based on device (irrespective of the designs). Where, the dynamic power utilization is obtained based on the design operation (In this work-PRESENT-80/128/256). The static power is utilization for the Artix-7 FPGA is fixed 0.083 W. This static power includes leakage power of the device and not depends on design. Rest of the dynamic power varies from the 0.103–0.109 W as the clock frequencies increases. The proposed work, reduces chip resources (Slices. LUTs) to improves the dynamic power. The time complexity in FPGA implementation is considering based on the execution time (Latency) in terms of Clock cycles. The latency (clock cycles) used for the Proposed PRESENT is mentioned in Tab. 6.

3.2 PRESENT-80/128/256 Encryption Module

Like the encryption module, the decryption module receives 64-bit ciphertext as an input and an 80/128/256-bit updated key as the input key. The detailed hardware architecture of PRESENT-80/128/256 decryption is represented in Fig. 3. It is an entirely reverse process of encryption and few changes in the SP-Network.

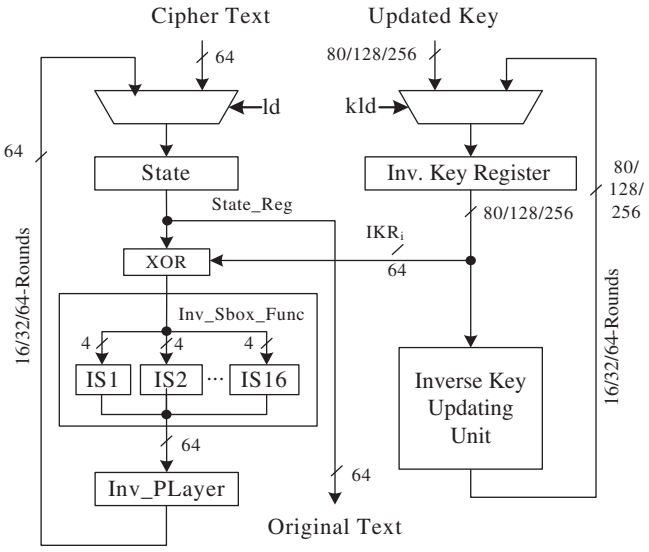

Figure 3: Hardware architecture of PRESENT-80/128/256 decryption module

The replacement of S-BOX, PLayer, and key updation module with Inv_Sbox_Func, Inv_PLayer [28], and inverse key updation unit, respectively. The Inverse Inv_Sbox_Func is tabulated in Tab. 3 and used in Inverse SP-network.

Table 3: PRESENT-80/128/256—Substitution Box (S-BOX) for decryption

In	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2
Out	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

3.3 Key Updation Modules for PRESENT-80/128/256

The key updation and inverse key updation architecture units are represented in Figs. 4a and 4b for PRESENT-80 encryption and decryption. For PRESENT-80, the key register (KR_i) stores 80-bit Data temporarily for 64-bit round key operation $KR_i = k_{63}k_{62}\ldots k_1k_0$ and starts updating. Consider the KR_i for PRESENT-80 module in Eq. (2) as follows:

$$KR_i = k_{63}k_{62}\ldots k_1k_0 ==> k_{79}k_{78}\ldots k_{17}k_{16} \tag{2}$$

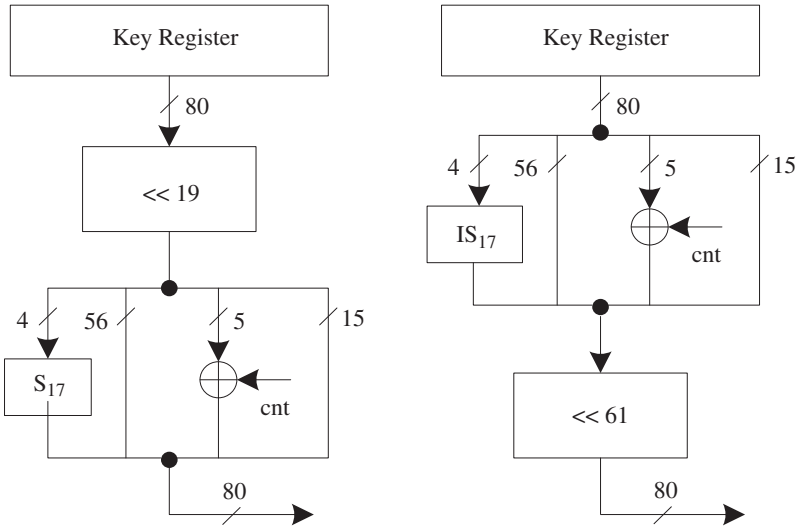

Figure 4: Hardware architecture of PRESENT-80 key modules (a) key updation (b) inverse key updation module

The 80-bit register key (KR_i) in Eq. (2) is mainly used to update the MSB portion of the first 64-bit key out of the 80-bit key for round operations in the Encryption process. Later perform the shifts left (<<) operations 19 times and further process the key updation using a round counter. The concatenation of the first 15-bits [14:0] keeps the same followed by the next 5-bits [19:15] are XORed with 5-bit round counter (cnt), and 56-bits [75:20] keeps the same, along with last 4-bits [79:76] are performed by S-BOX (S17) operation to generates the 80-bit updated key of PRESENT-80 Encryption. Similarly, for PRESENT-80 Decryption, the 80-bit key register data concatenates the First 15-bits [14:0] followed by the next 5-bits [19:15] are XORed with 5-bit round counter (cnt), 56-bits [75:20] keeps same, along with

last 4-bits [79:76] are performed by Inverse S-box (IS17) for the generation of the 80-bit key. This 80-bit key is left-shifted by 61 times for the generation of the 80-bit updated key.

The key updation and inverse key updation architecture units for PRESENT-128/256 encryption and decryption are similar to PRESENT-80 with few changes. For PRESENT-128 Encryption, the key register stores 128-bit Data temporarily and operate updating as follows. The 128-bit register key shifts left (<<) 67 times initially, then process the key updation. The 5-bits [66:62] are XORed with a 5-bit round counter (cnt), and MSB 8-bits [127:120] are performed by two S-BOX (S_{17} and S_{18}) operation for the generation of the 128-bit updated key. The other bits keep as it is with corresponding positions. Similarly, for PRESENT-128 Decryption, the 128-bit key register data uses 5-bits [66:62] are XORed with 5-bit round counter (cnt), and MSB 8-bits [127:120] are performed by two Inverse S-box (IS_{17} and IS_{18}) operation for the generates the 128-bit key. The rest of the bits keeps as it is with corresponding positions. This 128-bit key is left-shifted (<<) by 61 times to generate the 128-bit updated key.

The key updation and inverse key updation process for PRESENT-256 encryption and decryption as follows: For PRESENT-256 Encryption, the key register stores 256-bit Data temporarily and starts updating as follows. The 256-bit register key shifts left (<<) 195 times initially, then process the key updation using a round counter. The 5-bits [128:124] are XORed with a 5-bit round counter (cnt), and MSB 8-bits [255:248] are performed by two S-BOX (S_{17} and S_{18}) operation for the generation of the 256-bit updated key. Similarly, for PRESENT-256 Decryption, the 256-bit key register data uses 5-bits [128:124] are XORed with 5-bit round counter (cnt), and MSB 8-bits [255:248] are performed by two Inverse S-box (IS_{17} and IS_{18}) operation for the generates the 256-bit key. Finally, this 256-bit key is left-shifted (<<) 61 times to generate the 256-bit updated key.

4 Results and Analysis

The PRESENT-80/128/256 light-weight block ciphers for IoT security are designed using Verilog-HDL on the Xilinx Platform and implemented on the Artix-7 FPGA device for prototyping purposes. The simulation is carried out using the Modelsim 6.5f simulator. The PRESENT-80/128/256 modules of hardware constraints and comparative analysis with similar and different existing approaches are tabulated in this section.

The complete integration of encryption and decryption PRESENT-80/128/256 modules Performance results on Artix-7 FPGA are tabulated in Tab. 4. The Table contains area, frequency, total power utilization, and hardware efficiency with different key sizes and rounds for PRESENT-80/128/256 Modules. For example, the area utilization is <2%, and operating frequencies are also better on Artix-7 for PRESENT-80/128/256 modules with different rounds.

Table 4: Performance results of PRESENT-80/128/256 modules on Artix-7 FPGA

Resources	PRESENT-80			PRESENT-128			PRESENT-256		
Data size	64			64			64		
Key size	80			128			256		
Rounds	16	32	64	16	32	64	16	32	64
Area									
Slice registers	358	359	359	454	455	455	710	711	711
Slice LUTs	435	437	437	487	489	489	615	617	617
LUT-FF pairs	289	291	291	341	343	343	469	471	471

(Continued)

Table 4 (continued)

Resources	PRESENT-80			PRESENT-128			PRESENT-256		
Frequency									
Max. frequency (MHz)	410.79	410.7	412.4	410.79	410.79	412.49	410.79	410.79	412.4
Power									
Total power (W)	0.186	0.187	0.188	0.188	0.189	0.19	0.19	0.191	0.192
Hardware efficiency									
Throughput (Mbps)	1644	822	413	1644	822	413	1644	822	413
Throughput/Slice (Mbps/slice)	4.59	2.29	1.15	3.62	1.8	0.9	2.32	1.15	0.58
Throughput/LUTs (Mbps/LUT)	3.77	1.88	0.95	3.37	1.68	0.85	2.67	1.33	0.67

The PRESENT-80/128/256 consumes less total power with the range 0.186–0.192 W. Therefore, the power utilization is relatively more minor, and we can easily use these PRESENT modules for IoT devices for authentication purposes. The Throughput (Mbps) and efficiencies like Throughput/Slice (Mbps/slice) and Throughput/LUTs (Mbps/LUT) are also tabulated. The PRESENT-80/128/256 Module's Chip area (Slices and LUTs) and frequency utilization on Artix-7 is graphically represented in Fig. 5.

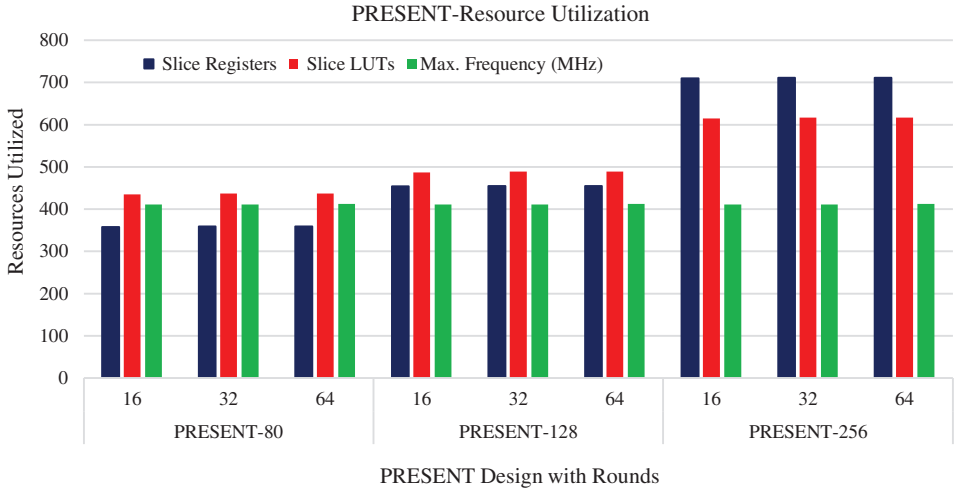

Figure 5: Resource utilization Graphical representation of PRESENT-80/128/256 module on Artix-7

The total power consumption of PRESENT-80/128/256 module with different clock frequencies on Artix-7 FPGA is represented in Fig. 6. The total power (W) mainly includes quiescent (Static) and dynamic power. The total power is generated using the Xpower analyzer tool on Xilinx Environment. The dynamic power varies with the PRESENT-80/128/256 design module. The static power is constant and does not depend on the design module, and depends on the selected device (Artix-7 FPGA). The static power for the PRESENT-80/128/256 is an average of 0.083 W, and dynamic power varies based on design modules. As the device clock frequencies increase, the total power of the PRESENT-80/128/256 is also increased.

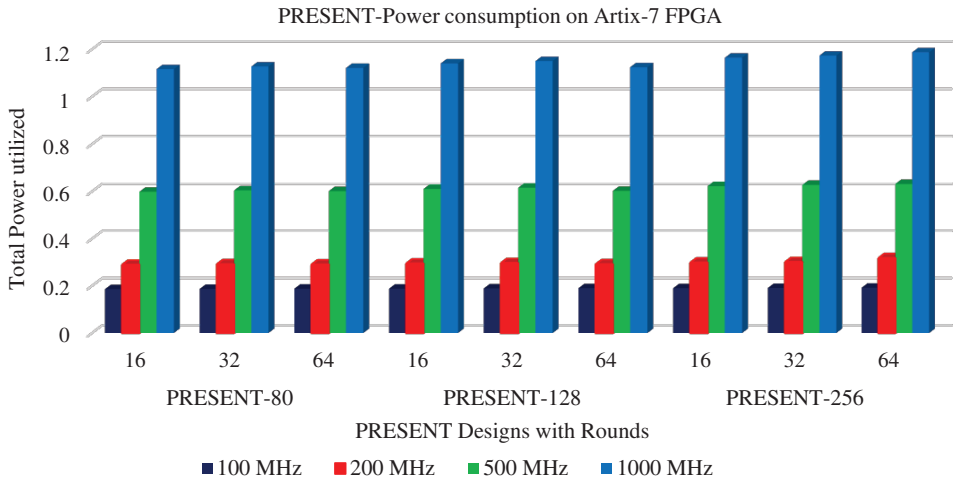

Figure 6: Power Consumption of PRESENT-80/128/256 Module with clock frequencies on Artix-7

The performance analysis of the proposed PRESENT-80/128/256 with different lightweight-block ciphers for 32-rounds is represented in Fig. 7 and tabulated in Tab. 5 with improvements.

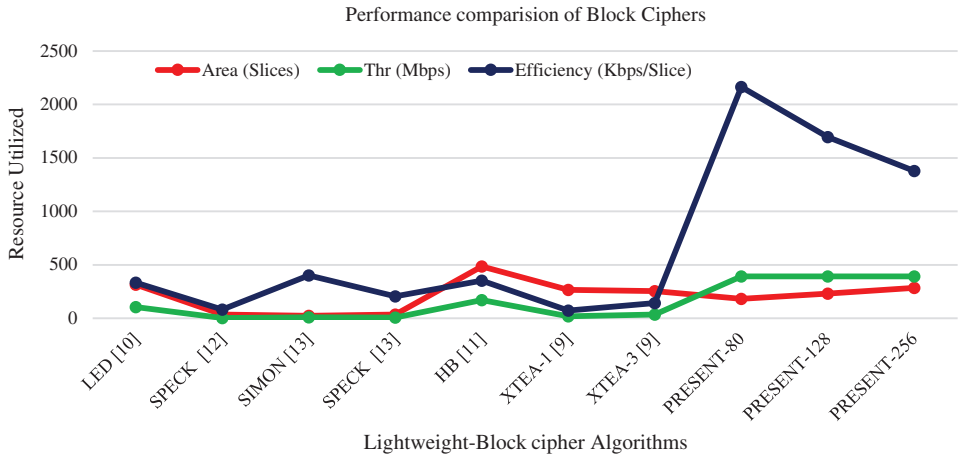

Figure 7: Performance comparison of PRESENT-with existing light-weight Block Ciphers

Table 5: Performance analysis of different lightweight-block ciphers with proposed work

Designs	Data size	Key	Area (Slices)	Throughput (Mbps)	Efficiency (Kbps/ Slice)	FPGA device
LED [10]	64	128	315	105.34	334.41	XC3S5000
SPECK [12]	128	128	34	2.8	82.35	XC3S50
SIMON [13]	64	128	24	9.6	400	XC3S50
SPECK [13]	64	128	34	7	205.88	XC3S50
HB [11]	16	64	485	171.36	353.3	XC3S200
XTEA-1 [9]	64	128	266	19	71.42	XC3S50

(Continued)

Table 5 (continued)

Designs	Data size	Key	Area (Slices)	Throughput (Mbps)	Efficiency (Kbps/ Slice)	FPGA device
XTEA-3 [9]	64	128	254	36	141.73	XC3S50
This work	64	80	181	391.54	2163.23	XC3S50
This work	64	128	231	391.54	1694.97	XC3S50
This work	64	256	284	391.54	1378.66	XC3S50

The different lightweight block ciphers include Tiny XTEA [9], LED [10], SPECK [12,13], SIMON [13], Hummingbird [11], are implemented on low-end Spartan-3 FPGA devices. The proposed work is compared with existing light-weight block ciphers with improved throughput (Mbps) and efficiency (kbps/slice) in the greater margin. These light-weight block ciphers consume more Latency for execution and affect overall performance in terms of Throughput and Efficiency.

The analysis of PRESENT-80/128 ciphers with similar recent PRESENT cipher modules are compared with different FPGA families like Spartan-6, Virtex-4, Virtex-5, Artix-7 are tabulated in Tab. 6 with improvements.

Table 6: Comparative analysis of PRESENT-80 and 128 block ciphers with recent existing works

Designs	Year	Data size	Key	FF's	LUT's	Slices	Fmax (MHz)	Lat (CC)	Thr (Mbps)	Thr* (Mbps)	*Efficiency (Kbps /Slice)
XC6slx16-3csg324											
Ref. [29]	2016	64	80	136	229	74	221.63	33	429.83	26.3	355.38
Ref. [30]	2017	64	128	201	220	61	210.66	136	99.13	6.38	104.61
Ref. [30]	2017	64	80	153	170	48	257.4	133	123.86	6.53	135.94
This work	2020	64	80	178	266	178	264.24	32	528.48	27.12	152.35
This work	2020	64	128	226	318	226	264.24	32	528.48	27.12	120.1
xc5vlx50t-3ff1136											
Ref. [30]	2017	64	80	153	190	67	542.3	133	260.96	6.53	97.39
Ref. [30]	2017	64	128	201	239	73	431.78	136	203.19	6.38	87.41
Ref. [31]	2019	64	80	149	216	54	545.05	51	683.98	17.01	315
Ref. [31]	2019	64	128	197	267	72	433.78	63	440.66	13.77	191.25
This work	2020	64	80	183	272	183	455.76	32	911.53	27.12	148.19
This work	2020	64	128	231	324	231	455.76	32	911.53	27.12	117.4
xc4vlx25-12ff668											
Ref. [30]	2017	64	80	153	215	124	375.66	133	180.77	6.53	52.62
Ref. [30]	2017	64	128	202	265	152	364.56	136	176	6.38	41.98
This work	2020	64	80	183	292	179	389.98	32	779.97	27.12	151.5
This work	2020	64	128	231	344	209	389.98	32	779.97	27.12	129.76

(Continued)

Table 6 (continued)

Designs	Year	Data size	Key	FF's	LUT's	Slices	Fmax (MHz)	Lat (CC)	Thr (Mbps)	Thr* (Mbps)	*Efficiency (Kbps /Slice)
xc5vlx110t-1ff1136											
Ref. [21]	2018	64	80	137	348	126	215.42	33	417.79	26.29	208.65
Ref. [21]	2018	64	128	137	396	150	212.13	33	411.41	26.29	175.26
This work	2020	64	80	183	272	183	455.76	32	911.53	27.12	148.19
This work	2020	64	128	231	324	231	455.76	32	911.53	27.12	117.4
xc7a100-1csg324											
Ref. [23]	2019	64	80	213	426	213	368	32	736	27.12	127.32
This work	2020	64	80	178	266	178	410.79	32	821.58	27.12	152.35

Note: * Denotes frequency of 13.56 MHz.

5 Conclusion and Future Work

The light-weight efficient, low-latency, High Throughput PRESENT-80/128/256 integrated Encryption, and Decryption modules are designed and implemented on FPGA for prototyping. The PRESENT80/128/256 Encryption and Decryption modules have their state updation and key updation units. The proposed work performs 16/32/64 rounds of state and key updation operations to create more confusion and diffusion to the attackers. The PRESENT-80/128/256 uses parallel operation for data and key updation, which consumes less execution time with better Latency. The PRESENT-80/128/256 is well suited to low-end IoT devices for security incorporation because of low Latency and High Throughput. The results are analyzed and compared with recent PRESENT models with improved chip area, Latency, Throughput, and efficiency on different FPGA devices. The PRESENT-80/128 is also compared with different light-weight algorithms with improved hardware constraints. The PRESENT-80/128 works with a throughput of 821.58 Mbps for 32-rounds on Artix-7 FPGA. In the future, Optimize the power performance by using the ASIC platform and also analyze the real-time power usage in IoT applications. In addition to that, analyze the critical attacks using PRESENT cipher on IoT infrastructure and its application to realize the security aspects.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] M. Frustaci, P. Pasquale, A. Gianluca and F. Giancarlo, "Evaluating critical security issues of the IoT world: Present and future challenges," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2483–2495, 2017.

[2] V. Hassija, C. Vinay, S. Vikas, J. Divyansh, P. Goyal *et al.,* "A survey on IoT security: Application areas, security threats, and solution architectures," *IEEE Access*, vol. 7, no. 5, pp. 82721–82743, 2019.

[3] S. Sezer, "T1c: IoT security: - Threats, security challenges and IoT security research and technology trends," *IEEE International System-on-Chip Conference (SOCC)*, vol. 7, no. 4, pp. 1–2, 2018.

[4] S. Sambit, S. Das and D. Swapan, "A new healthcare diagnosis system using an IoT-based fuzzy classifier with FPGA," *Journal of Supercomputing*, vol. 76, no. 8, pp. 5849–5861, 2020.

[5] F. Meneghello, C. Matteo, Z. Daniel, Michele, Polese and A. Zanella, "IoT: Internet of threats? A survey of practical security vulnerabilities in real IoT devices," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8182–8201, 2019.

[6] V. Arumugam, "Business Aspects, Models, and Opportunities of IoT," in *Edge Computing and Computational Intelligence Paradigms for the IoT, IGI Global*, USA, pp. 69–99, 2019.

[7] S. Satpathy, S. Debbarmaa and K. D. Bhattacaryya Bidyut, "Design a FPGA, fuzzy based, insolent method for prediction of multi-diseases in rural area," *Journal of Intelligent & Fuzzy Systems*, vol. 37, no. 5, pp. 7039–7046, 2019.

[8] B. Trim Steve and M. Jason, "FPGA security: From features to capabilities to trusted systems," in *Proc. of the 51st Annual Design Automation Conf.,* San Francisco, CA, USA, ACM, vol. 30, no. 3, pp. 1–4, 2014.

[9] Kaps and Jens-Peter, "Chai-tea, cryptographic hardware implementations of XTEA," in *Int. Conf. on Cryptology*, Kharagpur, India, Springer, vol. 131, no. 5, pp. 363–375, 2008.

[10] R. RajaRaja and D. Pavithra, "Implementation of hardware efficient light-weight encryption method," in *Int. Conf. on Communication and Signal Processing*, Melmaruvathur, India, vol. 2, no. 39, pp. 191–195, 2013.

[11] T. Hari Krishnan and C. Babu, "Cryptanalysis of hummingbird algorithm with improved security and throughput," in *Int. Conf. on VLSI Systems, Architecture, Technology, and Applications (VLSI-SATA)*, Bengaluru, India, vol. 5, no. 15, pp. 1–6, 2015.

[12] A. Nemati, F. Soheil, A. Arash and M. Vahab Al-din, "A low-cost and flexible FPGA implementation for SPECK block cipher," in *12th Int. Iranian Society of Cryptology Conf. on Information Security and Cryptology (ISCISC)*, Rasht, Iran, vol. 7, no. 7 pp. 42–47, 2015.

[13] R. Beaulieu, S. Douglas, S. Jason, S. Treatman-Clark, B. Weeks *et al.,* "SIMON and SPECK: Block ciphers for the Internet of Things," *IACR Cryptology*, vol. 15, no. 175, pp. 1–15, 2015.

[14] S. P. Guruprasad and B. S. Chandrasekar, "An evaluation framework for security algorithms performance realization on FPGA," in *IEEE Int. Conf. on Current Trends in Advanced Computing (ICCTAC)*, Bangalore, India, vol. 1, no. 5, pp. 1–6, 2018.

[15] R. Bharathi and N. Parvatham, "LEA-Siot: Hardware architecture of lightweight encryption algorithm for secure IoT on FPGA platform," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 1, pp. 720–725, 2020.

[16] R. Bharathi and N. Parvatham, "Hardware-based physical layer security solutions and algorithms for IoT devices on FPGA platform," *International Journal of Innovative Technology and Exploring Engineering*, vol. 9, no. 3, pp. 2128–2132, 2020.

[17] J. B. Mohd, H. Thaier and V. V. Athanasios, "A survey on lightweight block ciphers for low-resource devices: Comparative study and open issues," *Journal of Network and Computer Applications*, vol. 58, pp. 73–93, 2015.

[18] G. Edwar, H. Cesar and M. Fredy, "Performance evaluation of the present cryptographic algorithm over FPGA," *Contemporary Engineering Sciences*, vol. 10, no. 12, pp. 555–567, 2017.

[19] H. Azari and V. J. Prashant, "An efficient implementation of present cipher model with 80 bit and 128 bits key over FPGA based hardware architecture," *International Journal of Pure and Applied Mathematics*, vol. 119, no. 14, pp. 1825–1832, 2018.

[20] C. G. Thorat and V. S. Inamdar, "Implementation of new hybrid lightweight cryptosystem," *Applied Computing and Informatics*, vol. 16, no. 1/2, pp. 195–206, 2018.

[21] L. Chom Thungon, N. Ahmed and I. Hussain, "Comparison of AES and PRESENT block cipher for 6LoWPAN based internet-of-things," *International Journal of Computational Intelligence & IoT*, vol. 1, no. 2, pp. 255–259, 2018.

[22] R. Aragona, C. Marco, T. Antonio and T. Maria, "Primitivity of PRESENT and other lightweight ciphers," *Journal of Algebra and Its Applications*, vol. 17, no. 6, pp. 1850115–16, 2018.

[23] C. Pei, X. Yang, L. Wei and H. Xiaojia, "Trade-off of security and performance of lightweight block ciphers in industrial wireless sensor networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 117, no. 1, pp. 1–18, 2018.

[24] M. Jangra and S. Buddha, "Performance analysis of CLEFIA and PRESENT lightweight block ciphers," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 22, no. 8, pp. 1489–1499, 2019.

[25] B. Rashidi, "Flexible structures of lightweight block ciphers PRESENT, SIMON and LED," *IET Circuits, Devices & Systems*, vol. 14, no. 3, pp. 369–380, 2020.

[26] R. Jenny, R. Sherine, R. Sudhakar and M. Karthikpriya, "Design of compact S Box for resource-constrained applications," *Journal of Physics: Conference Series*, vol. 1767, no. 1, pp. 012059, 2021.

[27] K. Jang, S. Gyeongju, K. Hyunjun, K. Hyeokdong, K. Hyunji *et al.,* "Efficient implementation of PRESENT and GIFT on quantum computers," *Applied Sciences*, vol. 11, no. 11, pp. 4776, 2021.

[28] A. Bogdanov, R. Lars, L. Gregor, P. Christof *et al.,* "PRESENT: An ultra-lightweight block cipher," in *Int. Workshop on Cryptographic Hardware and Embedded Systems*, Vienna, Austria, Springer, vol. 765, no. 3, pp. 450–466, 2007.

[29] Lara-Nino, A. Carlos, M. Morales-Sandoval and A. Diaz-Perez, "Novel FPGA-based low-cost hardware architecture for the PRESENT block cipher," in *IEEE Euromicro Conf. on Digital System Design (DSD)*, Limassol, Cyprus, vol. 1, no. 85, pp. 646–650, 2016.

[30] Lara-Nino, A. Carlos, A. Diaz-Perez and M. Morales-Sandoval, "Light-weight hardware architectures for the present cipher in FPGA," *IEEE Transactions on Circuits and Systems*, vol. 64, no. 2, pp. 2544–2555, 2017.

[31] V. K. Reddy, S. Rao, R. Akhil, and P. Sathish Kumar, "FPGA implementation of present algorithm with improved security," in *IEEE 3rd Int. Conf. on Electronics, Communication, and Aerospace Technology (ICECA)*, Coimbatore, India, vol. 2, no. 79, pp. 404–409, 2019.