International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 06 Issue: 06 | June - 2022 Impact Factor: 7.185 ISSN: 2582-3930

IMPLEMENTATION OF HARDWARE EFFICIENT LIGHT WEIGHT ENCRYPTION METHOD USING ITERATIVE ARCHITECTURE

Santhosh Kumar S, Sethuraman N, Vishal V
Electronics and Communication Engineering, Velammal College of Engineering and Technology, Madurai.

***Abstract*— This paper proposes to implement a hardware efficient light weight encryption algorithm based on Light Encryption Device (LED). The hardware efficiency of a Light Encryption Device (LED) is mainly determined by the implementation of the Mix Columns operation. In order to reduce the computations involved, the. An iterative architecture is used to implement the designed LED algorithm, so that the hardware elements can be reused for every round operation. Further Block RAMs (BRAMs) are utilized for reducing area utilization. The proposed work is done for the block size of 64 bits and the key size of 128 bits and is targeted to Spartan 3E.**

***Index Terms*—BRAMs, Data Security, Light weight cryptography, VLSI.**

I. INTRODUCTION

Information security has become a top priority in many applications such as smart cards, Radio Frequency Identification (RFID) tags and sensor nodes. All these applications require the access of private information. For example, in electronic transactions, customers provide credit card or debit card numbers when making payments online. If the connection is insecure, hackers can easily access this sensitive information [13]. Hence, there is a great demand for cryptographic algorithms. In order to provide secure communication of data the algorithm must ensure that

1. The data is kept secret,
2. The entirety of data is preserved,
3. The identity of the users is genuine, and
4. The users cannot deny the transmission.

Cryptographic algorithms are used to ensure confidentiality and it comes within one of two categories: symmetric-key and asymmetric-key. Symmetric-key algorithms use the same key for both encryption and decryption. Asymmetric-key algorithms use a public key for encryption and a private key for decryption. Block ciphers and stream ciphers are two types of symmetric-key algorithms. Block ciphers encrypt a block ofdata whereas stream ciphers encrypt the data bit by bit [13].

In cryptology, block ciphers are one of the most important primitives. A block cipher consists of two closely related algorithms, viz, block encryption and block decryption algorithms.

Block encryption algorithm takes an input block of fixed-size known as the plaintext and converts it into another block of the same size known as the cipher text. This block encryption and decryption takes place under the action of a fixed secret key that may not have the same size of the plain text. A block cipher must be invertible i.e., by using the block decryption algorithm it should be possible to recover the original plaintext from the cipher text and the secret key. Once the plaintext has been encrypted using a given key, a successful decryption can be performed only by knowing the secret key [10].

The pervasive computing will be characterized by many smart devices. These compact devices have very limited resources in terms of memory, computing resources and battery due to tight cost constraints [11]. In order to provide security in these resource constrained devices, light weight cryptographic algorithms have been developed. Today, many light weight block ciphers such as PRESENT, KLEIN, etc., provide the means of light weight cryptography. Among all these, LED block cipher stays ahead in terms of security even without key schedule. LED block cipher is more resistant to classical attacks and also to the related key attacks when compared to other existing light weight block ciphers [4].

As the data rates continue to increase, software based security systems are inadequate and hardware-efficient algorithms become more significant. However, there has not been much research on porting these light weight algorithms to FPGAs. The design trade-offs in implementing light weight algorithms in hardware is shown in Fig.1. In this paper we propose a hardware efficient FPGA implementation of Light Encryption Device algorithm by optimizing area and speed. The design is targeted to Xilinx Spartan3 FPGA.

The rest of this paper is organized as follows. Section II gives a brief overview of a LED Block Cipher. Section III describes the round operations of the LED algorithm. The proposed method of hardware implementation of LED algorithm is explained in Section IV. The iterative architecture is described in Section V. Section VI demonstrates the simulation results and the conclusion is drawn in Section VII.

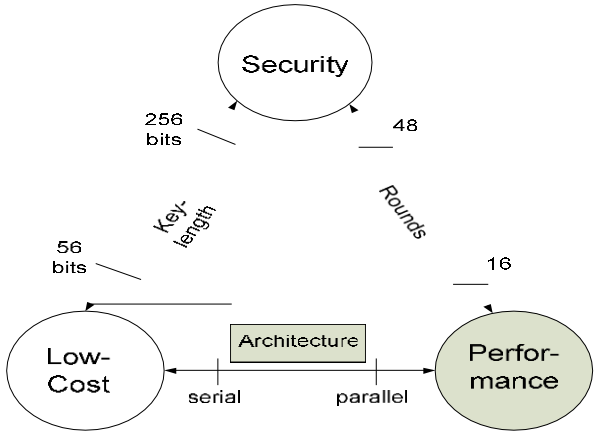

Fig. 1. Design Trade-offs for light weight cryptography [12].

II. Light Encryption Device (LED) Block Cipher: A Brief Overview

Light Weight Encryption Device block cipher is a 64 bit block cipher that uses cryptographic key sizes varying from 64 bits to 128 bits [4]. In this paper, 128 bit key size is used and it has 48 rounds of operation.

Fig.2 shows the overall view of the algorithm Light Encryption Device for the key size of 128 bits. Here there are totally 48 rounds of operation is involved. Initially Plain text (P) is EX-OR'ed with Key1 (K1) where K1 ranges from 0 to 63 bits of total key size. After the completion of first four rounds Key2 (K2) is EX-OR'ed, where K2 ranges from 64 to 127 bits of the total key size. K1and K2 are alternatively EX-OR'ed after every four rounds. The four rounds of operation is called a step.

Internally, operations of LED algorithm are performed on a two-dimensional array of nibbles called the state. The State consists of four rows of nibbles. The data blocks are internally represented in 4X4 arrangement, called state.

III. Operations Of LED Algorithm

The Light Encryption algorithm is described by four main operations. They are,

1. Add Constants
2. Substitute Cells
3. Shift Rows
4. Mix-Columns serial

Fig.3 shows the operations which are involved in a single round of Light Weight Encryption Device. The cipher state is conceptually arranged in a (4X4) grid where each nibble represents an element from Galois Field GF (2^4) with the underlying polynomial for field multiplication is given by (x^4+x+1). Each element of the matrix is of 4 bits.

A. Add Round Key

The Add Round Key operation combines nibbles of K1 or K2 with the state, corresponding array positioning, using bitwise Exclusive-OR. In ADD Constants a round key is added to the state by a simple bit-wise XOR operation. There is no key schedule, or rather this is the sum total of the key schedule, and the arrays K1 and appropriate K2 are alternatively used without modification [4].

B. Add Constants

At each round the 6 bits (rc_5,rc_4,rc_3,rc_2,rc_1,rc_0) are shifted one position to the left. The new value to rc_0 is computed as $rc_5$③$rc_4$③1. The 6 bits are initialized to zero and updated before they are used in every round. The values of the (rc_5,rc_4,rc_3,rc_2,rc_1,rc_0) constants for each round are given in Table I.

Table I values are encoded to nibble values for each round, with rc_0 being the least significant bit [4]. The constant, when used in each round, is arranged into an array as given in Eq.1.

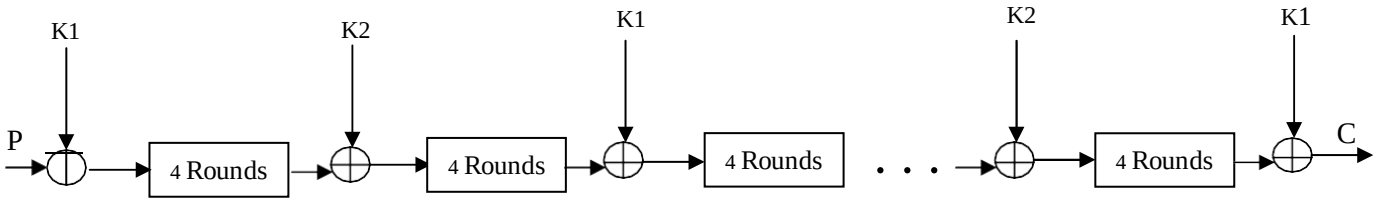

Fig. 2. Over-view of Light Encryption Device (LED) [4].

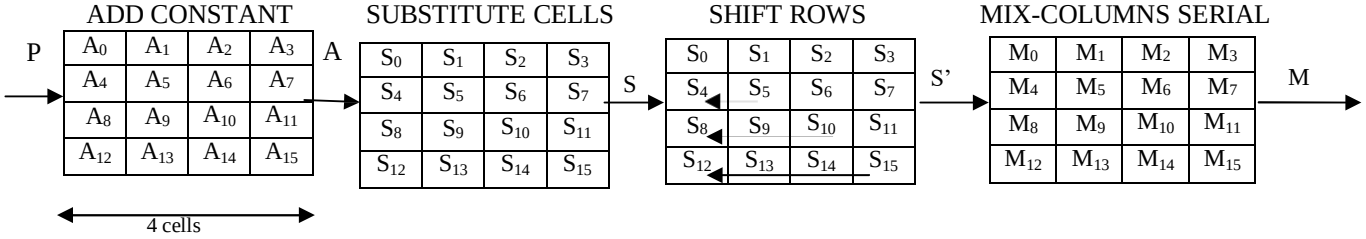

Fig. 3. Operations involved in a single round.

TABLE I
TABLE OF CONSTANTS OF EACH ROUND [4]

Rounds	Constants
1-12	01,03,07,0F,1F,3E,3D,3B,37,2F,1E,3C
13-24	39,33,27,0E,1D,3A,35,2B,16,2C,18,30
25-36	21,02,05,0B,17,2E,1C,38,31,23,06,0D
37-48	1B,36,2D,1A,34,29,12,24,08,11,22,04

$$A = \begin{bmatrix} 0 & (rc_5||rc_4||rc_3) & 0 & 0 \\ 1 & (rc_2||rc_{1||}rc_0) & 0 & 0 \\ 2 & (rc_5||rc_4||rc_3) & 0 & 0 \\ 3 & (rc_2||rc_1||rc_0) & 0 & 0 \end{bmatrix} \quad (1)$$

C. Substitute Cells

The Substitute Cell transformation is a non-linear nibble substitution that operates independently on each cell of the state using a substitution table (S-Box) [4]. The S-Box which is invertible is sixteen 4-bit hexadecimal values. Each element in the state array is replaced by the element from the S-BOX. The action of this Box in hexadecimal notation is given in Table

For S-Box the current state b_{63} ... b_0 is considered as sixteen4-bit words and it is represented $w_{15}...w_0$ as where $w_i=b_4*i+3"b_4*i+2"b_4*i+1"b_4*i$ for $0\leq i \leq 15$ and the output nibble $S[w_i]$ provides the updated state values [2].

D. Shift Rows

In the Shift Rows transformation, the nibbles in the last three rows of the state are cyclically shifted left over different number of nibbles. The first row is not shifted. The shift rows operation of LED algorithm is shown in Fig.4 in which S denotes the un-shifted matrix and S' denotes the shifted matrix.

E. Mix-Columns Serial

Each column of the array state is viewed as a column vector and is replaced by the column vector that results after multiplying the vector by the matrix J as in Eq.2.

The final value of the state provides the cipher text with nibbles of the array being unpacked in the same way. Fig.5 shows the Mix-Columns Serial matrix operation of LED algorithm.

$$J = \begin{bmatrix} 0&1&0&0 \\ 0&0&1&0 \\ 0&0&0&1 \\ 4&1&2&2 \end{bmatrix}^{\wedge 4} = \begin{bmatrix} 4&1&2&2 \\ 8&6&5&6 \\ B&E&A&9 \\ 2&2&F&B \end{bmatrix} \quad (2)$$

The equations (3), (4), (5) and (6) represent the approach for Mix-Columns Serial operation involved in Fig.5 for one column.

TABLE II
S-BOX OF LED [4]

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

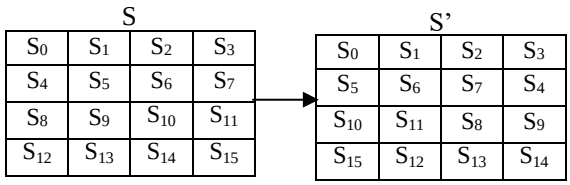

Fig. 4. Shift rows operation of LED algorithm

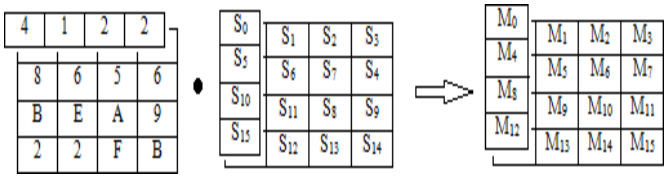

Fig. 5. Mix-columns operation of LED algorithm.

The dot operator ($\bullet$) in Fig.5 as well as in the following equations represents the Galois Field Multiplication in (2^4).

$$M_0 = (4 \bullet S_0) \oplus (1 \bullet S_5) \oplus (2 \bullet S_{10}) \oplus (2 \bullet S_{15}) \quad (3)$$

$$M_4 = (8 \bullet S_0) \oplus (6 \bullet S_5) \oplus (5 \bullet S_{10}) \oplus (6 \bullet S_{15}) \quad (4)$$

$$M_8 = (B \bullet S_0) \oplus (E \bullet S_5) \oplus (A \bullet S_{10}) \oplus (9 \bullet S_{15}) \quad (5)$$

$$M_{12} = (2 \bullet S_0) \oplus (2 \bullet S_5) \oplus (F \bullet S_{10}) \oplus (B \bullet S_{15}) \quad (6)$$

IV. HARDWARE IMPLEMENTATION OF LED ALGORITHM (PROPOSED METHOD)

The efficiency of hardware implementations is mainly determined by area and speed of the algorithm. The hardware implementation of LED block cipher by Jian Guo, Thomas Peyrin, Axel Poschmann and Matt Robshaw has been proposed in 2011 and it is implemented using Mentor graphics [4]. Their work is based on serial hardware architecture of LED algorithm. Their serialized design consists of seven modules: Mix Columns Serial, state, Add Key, Add Constant, Substitute Cells, Controller and key state [4]. In their design it requires 39 clock cycles to perform one round of LED, resulting in a total latency of 1872 clock cycles for LED-128. In this paper the Mix Columns Serial operation of LED algorithm is implemented using an efficient GF (2^4) multiplier [13]. The multiplication operation for four bit value of matrix element '4' is given as,

$$4 \bullet A_{\circ} \rightarrow \begin{bmatrix} 0&0&1&0 \\ 0&0&1&1 \\ 1&0&0&1 \\ 0&1&0&0 \end{bmatrix} \bullet \begin{bmatrix} a0 \\ a1 \\ a2 \\ a3 \end{bmatrix} \quad (7)$$

But this method involves more computations i.e., multiplications in GF (2^4) for every round.

V. ITERATIVE ARCHITECTURE

The basic iterative architecture shown in Fig. 6 is designed to implement an encryption unit. This reduces the hardware utilization and the hardware is reused for every round. One round of the cipher is implemented as a combinational logic and supplemented with a multiplexer. In the first clock cycle, input block of data is fed to the circuit through the multiplexer. In each subsequent cycle, one round of cipher is evaluated and the result is fed back to the circuit through the multiplexer.

The two characteristic features of this architecture are only one block of data is encrypted at a time and the number of clock cycles necessary to encrypt a single block of data isequal to the number of cipher rounds [10].

TABLE III
TRANSFORMATION BOX

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2
2	B	A	C	5	1	0	7	9	6	F	D	3	8	E	2	4
3	0	F	A	E	8	0	D	0	0	1	2	B	C	0	3	6
4	5	7	B	A	2	0	E	1	C	D	9	6	3	F	4	8
5	9	2	D	1	B	0	4	C	F	3	C	E	7	8	5	A
6	E	D	7	F	3	0	9	8	A	2	4	5	B	1	6	C
7	0	8	1	4	A	0	3	0	0	C	B	D	F	0	7	E
8	A	E	5	7	4	0	F	2	B	9	1	C	6	D	8	3
9	6	B	3	C	D	0	5	F	8	7	E	4	2	A	9	1
A	1	4	9	2	5	0	8	B	D	6	C	F	E	3	A	7
B	D	1	F	9	C	0	2	6	E	8	3	7	A	4	B	5
C	0	9	E	D	6	0	6	0	0	4	8	A	5	0	C	B
D	0	C	8	6	F	0	6	0	0	A	7	2	1	0	D	9
E	4	3	2	8	7	0	6	A	1	B	5	9	D	C	E	F
F	8	6	4	3	E	0	C	7	2	5	A	1	9	B	F	D

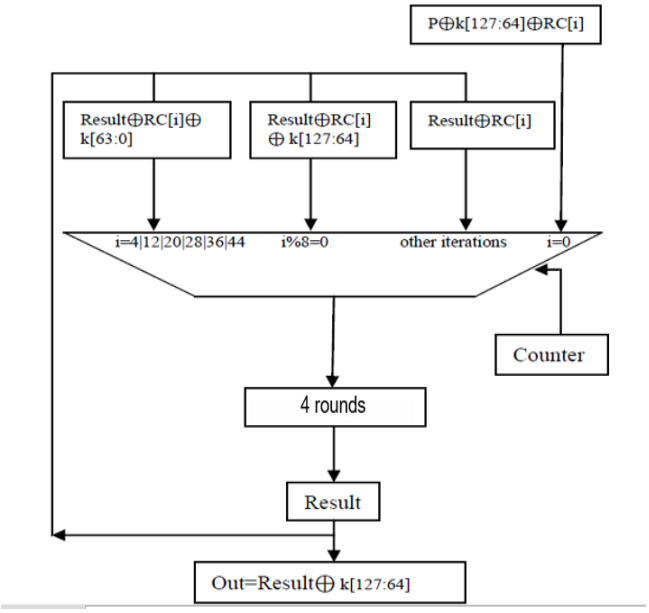

Fig. 6. Iterative architecture

For the first round (i=0) plain text is EX-OR'ed with the 64 to 127 bits of key and also EX-OR'ed with the constant RC[i] to get the next round's Add Constant (AC). For the rounds where i=4|12|20|28|36|44, the previous round result is EX-OR'ed with 0 to 63 bits of key and also with corresponding RC[i] to get next round's AC. For the rounds where (i mod 8) = 0, the previous round result is EX-OR'ed with 64 to 127 bits of key and also with corresponding RC[i]. For the other rounds of iteration the result of the previous round is EX-OR'ed with the corresponding RC[i] to obtain AC of the next immediate round. This AC is fed as input to the T-Box and the corresponding result from T-Box is fed to the multiplexer. At the end of the final round (i=47) result is EX-OR'ed with 64 to127 bits of key to get final encrypted output (cipher text).

VI. RESULTS AND DISCUSSION

The LED encryption algorithm is designed using verilog HDL and simulated using Xilinx ISim simulator. The algorithm is verified using appropriate test inputs as shown below (represented in Hexadecimal). The encryption module is simulated with a 64 bit input plain text and 128 bit key.

Plain text: 0123456789ABCDEF

Key: 0123456789ABCDEF0123456789ABCDEF

After 48 rounds of operation the cipher text obtained is,

Cipher text: 3131C231205C36

The results obtained are compared and verified with the test results [4].

The LED encryption module designed is targeted to Spartan 3 FPGA (xc3s5000). The design is synthesized using Xilinx XST synthesis tool. Area and speed are considered as major parameters of interest. The synthesis results are shown in Table IV. In the first approach the Mix Columns Serial operation of LED algorithm is implemented using a GF (2^4) multiplier and Substitute Cells operation is configured using slices of FPGA. The number of slices utilized by the Galois field multiplier approach is 315. Similarly the number of slice flip flops and the number of four input LUTs utilized by the Galois field multiplier approach is 70 and 556 respectively.

In the proposed method the Substitute Cells and Mix-Columns Serial operation are combined into a single transformation step

Since the LED algorithm has 48 rounds of operation for 128 bit key size, the hardware setup is used iteratively to get reduced area utilization.

The device utilization summary shows that LED Encryption using iterative architecture approach has reduced area utilization and improved speed when compared to LED Encryption using GF multiplier.

TABLE IV
V II. DEVICE UTILIZATION

Logic Utilization	Used	Available	Utilization
Number of Slices	321	960	33%
Number of slice flip flops	71	1920	3%
Number of 4 input LUTs	611	1920	31%
Number of bonded IOBs	257	66	389%
Number of GCLKs	1	24	4%

VIII. CONCLUSION AND FUTURE WORK

In this work, encryption of the block cipher LED is presented. The transformation box has been successfully derived by combining both the Substitute Cells and Mix-Column Serial step. An iterative architecture is designed for reduced area utilization. The encryption process of LED has been successfully simulated using ISim simulator and synthesized using XST synthesizer. The design is targeted to Spartan 3 device of FPGA.

This work is mainly intended to reduce area utilization and improve the speed of LED encryption. The other aspects of algorithm such as reducing the number of rounds by maintaining the provable security and key scheduling are not considered here. These aspects may be addressed for future work. Further improvements in speed can be obtained by using pipelined architecture.

.

REFERENCES

[1] Y. Gao, H. Ao, Z. Feng, W. Zhou, S. Hu, and W. Tang, “Mobile network security and privacy in WSN, Procedia Computer Science, 129, pp. 324-330, 2018.

[2] T. Abdelmoghni, O. Z. Mohamed, B. Billel, M. Mohamed, and L. Sidahmed, “Implementation of AES coprocessor for wireless sensor networks,” 2018 International Conference on Applied Smart Systems (ICASS), 2018.

[3] Y. Sverdlik. “The world’s 10 fastest supercomputer – in pictures,” Data Center Knowledge, 2019. [Online]. Available:https://www.datacenterknowledge.com/supercomputers/wo rld-s-10-fastes-supercomputers-pictures. [Accessed: 03-Aug-2019]

[4] D. Puthal, S. P. Mohanty, P. Nanda, E. Kougianos, and G. Das, ‘‘Proof-ofauthentication for scalable **blockchain** in resource-constrained distributed systems,’’ in Proc. IEEE Int. Conf. Consum. Electron. (ICCE), Las Vegas, NV, USA,

[5] Y. Su, Y. Gao, O. Kavehei, and D. C. Ranasinghe, ‘‘Hash functions and benchmarks for resource constrained passive devices: A preliminary study,’’ in Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PerCom Workshops), Kyoto, Japan, Mar. 2019, pp. 1020–1025.

[6] J. Kim, J. Cho, and D. Park, ‘‘Low-power command protection using SHA-CRC inversion-based scrambling technique for CAN-integrated automotivecontrollers,’’inProc.IEEEConf.DependableSecureComput. (DSC), Kaohsiung, Taiwan, Dec. 2018.

[7]M. Chamekh, M. Hamdi, S. El Asmi, and T.-H. Kim, ‘‘Security of RFID based Internet of Things applications: Requirements and open issues,’’ in Proc. 15th Int. Multi-Conf. Syst., Signals Devices (SSD), Hammamet, Tunisia, Mar. 2018.

[8] C.Krull,L.F.McMillan,R.M.Fewster,R.vanderRee,R.Pech,T.Dennis, and M. C. Stanley, ‘‘Testing the feasibility of wireless sensor networks and the use of radio signal strength indicator to track the movements of wild animals,’’ Wildlife Res., vol. 45, no. 8, pp. 659–667, Jan. 2019.

[9] L. Chen, K. Cong, and S. Sultana, ‘‘Side-channel attack detection using hardware performance counters,’’ U.S. Patent 16234085, May 2, 2019.

[10]J. A. Nix, ‘‘Cryptographic unit for public key infrastructure (PKI) operations,’’ U.S Patent 15575908, May 24, 2018.