International Journal of Control and Automation
Vol. 10, No. 1 (2017), pp.187-198
http//dx.doi.org/10.14257/ijca.2017.10.1.17

The Efficient Hardware Design of a New Lightweight Block Cipher

Gookyi Dennis A. N.[1], Seungyong Park[2] and Kwangki Ryoo[*]

Department of Information and Communication Engineering, Hanbat National University, Daejeon, 305-719, South Korea
[1]dennisgookyi@gmail.com, [2]srrr.kr@gmail.com, [*]kkryoo@gmail.com

Abstract

In this paper, we propose a new lightweight compact block cipher for pervasive computing. The cipher consists of a 128-bit key and uses 8 rounds to encrypt a block of 64-bit data. It makes use of the Feistel structure together with an S-Box and P-box. We implemented our cipher on iNEXT-V6 test board, which is equipped with virtex6 FPGA. The design synthesized to 196 slices at 337 MHz maximum clock frequency. The hardware results indicate that our cipher uses minimal hardware resources and has greater throughput as compared to other lightweight ciphers.

Keywords: *lightweight cipher, pervasive computing, Feistel, S-Box, P-Box, FPGA.*

1. Introduction

"The most profound technologies are those that disappear, they weave themselves into the fabric of everyday life until they are indistinguishable from it" [1]. This statement by Mark Weiser in his report in the early 90's ushered the computing paradigm into the era of pervasive or ubiquitous computing. As of today, there is a mass deployment of pervasive computing in almost all sectors of human livelihood. In pervasive computing, devices with computing capabilities are attached to household items to keep track of sensitive information about the host they are attached to. These devices record, store, and update sensitive information about their hosts. Because the information on these devices is sensitive, it is therefore sad to note that not much work has been done to protect the information stored on these devices. With the mass deployment of pervasive computing, it is inherent to use low cost devices to alleviate the cost of implementation. Low cost devices indicate that these devices are constrained in terms of computing capabilities, memory capacitance, and power supply. Radio Frequency Identification (RFID) Technology is widely used in pervasive computing. An RFID tag consists of an antenna and an Integrated Circuit (IC). The IC normally has a user memory of less than 512-bit, and the response time of the tag should be less than 100us according to the ISO/IEC 18000 standard [2]. With these constraints, implementing standard cryptographic algorithms like AES [3] on such a device is deemed impractical. The research focus has therefore shifted to lightweight cryptography. Most lightweight ciphers fall short of the ISO/IEC standard on lightweight ciphers, which mandate the number of LUT slices to be less than 300 [4]. Also, most lightweight ciphers use a large number of rounds which lead to low performance that requires over 100us to encrypt a block of data at 100 KHz. In this paper, we present the design and implementation of an entirely new ultra-lightweight block cipher that is both fast and efficient in hardware.

ISSN: 2005-4297 IJCA

1.1. Features and Design Principle

We propose a new ultra-lightweight block cipher. The cipher has a key size of 128-bit and uses 8 rounds to encrypt a data block of 64-bit. Some of the design decisions that were considered in the design of the cipher include:

- The Feistel structure makes the encryption and decryption routines basically the same. This leads to the implementation of only one core for both encryption and decryption
- The use of a 128-bit key size provides a high level of security because it discourages the mounting of a brute force attack
- The only mathematical operator used in the cipher is the XOR operator, which leads to the use of minimal hardware resources for implementation
- To prevent a backdoor into the cipher, the substitution box (S-Box) of PRESENT [5] algorithm was used. This S-Box is extremely lightweight and resistant to linear and differential cryptanalysis
- A completely new permutation box (P-Box) was designed. This is the Key Dependent One Stage Omega Permutation P-Box. This provides further security and leads to better permutation of bits
- The key schedule algorithm involves only a shift operation leading to the use of minimal hardware resources and also on-the-fly computation of round keys.

1.2. Cipher Comparison

A number of lightweight ciphers have been proposed for pervasive computing: HIGHT, PRESENT [6], CLEFIA [7], TEA, DESL and LEA. Table 1 shows the comparisons of these algorithms. Two of these algorithms (PRESENT and CLEFIA) have been accepted as ISO/IEC standards.

- PRESENT cipher is a 128-bit key size, 64-bit block size and a 31 round cipher. It has one of the smallest S-Boxes available. The downside to PRESENT algorithm is that it requires 31 rounds to encrypt a block of data. This translates to 310us per block at 100 KHz. This falls short of the 100us response time for RFID tags.
- CLEFIA cipher is a 128-bit block cipher that uses 128, 192 or 256-bit key and requires 18, 22 or 26 rounds depending on the key size. CLEFIA uses two S-Boxes and two P-Boxes and therefore requires more than the stipulated 512-bit of memory reserved in RFID tags.

Table 1. Lightweight Ciphers

Algorithm	S-Box Memory requirement (bits)	Cycles/block	Time for 1 encryption @100KHz (us)
PRESENT	64	31	120
TEA	-	64	640
HIGHT	-	32	320
CLEFIA	4096	18	180
LEA	-	24	240
Proposed	64	8	80

2. Design of New Lightweight Cipher

The design of the proposed cipher is tailored toward high efficiency and moderate security. In terms of efficiency, we simulated various ciphers and compared their real time processing speed to our proposed cipher. Lightweight ciphers trade minimal hardware resources and performance for security. Here we describe the design of the proposed cipher.

2.1. State Representation

Considering a 128-bit user input key, let key [127:0] represent the 128-bit key, let k[0], k[1], …, k[15] represent an array of 8-bits from key[127:0]. The key [127:0] is represented as four 32-bit words K[0], K[1], K[2] and K[3]. Each k[i] is taken from the key[127:0] as follow:

$$k[i] = key[8i+7 : 8i] \text{ for } 15 \leq i \leq 0 \quad (1)$$

Each K[i] is taken from the bytes k[i] as follows:

$$K[i] = k[4i+3] \,||\, k[4i+2] \,||\, k[4i+1] \,||\, k[4i] \text{ for } 3 \leq i \leq 0 \quad (2)$$

Considering the user input 64-bit data, let state[63:0] represent the 64-bit data, let s[0], s[1], …, s[7] represent an array of 8-bits from state[63:0]. The state[63:0] is represented as two 32-bit words S[0] and S[1]. Each s[i] is taken from state[63:0] as follows

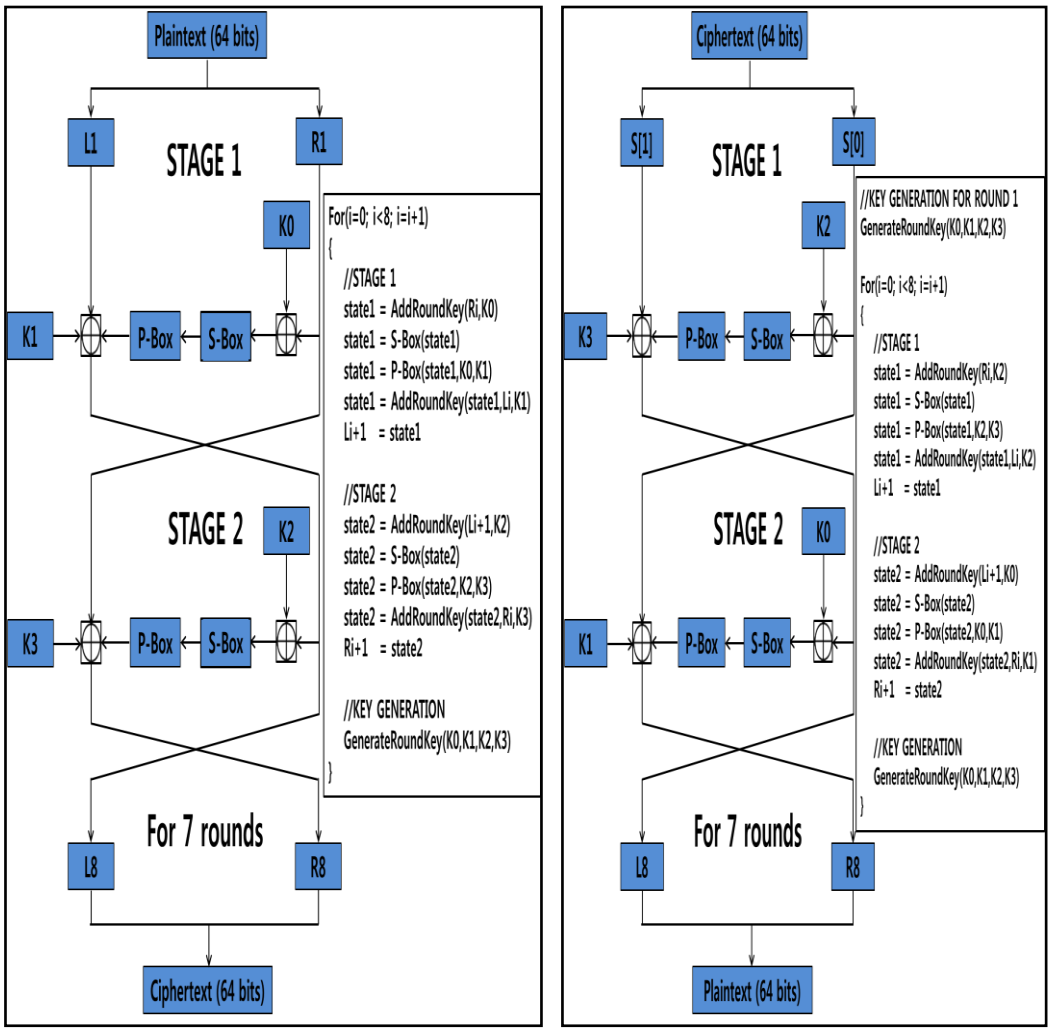
$$s[i] = state[8i+7 : 8i] \text{ for } 7 \leq i \leq 0 \quad (3)$$

Each S[i] is taken from the bytes s[i] as follows

$$S[i] = s[4i+3] \,||\, s[4i+2] \,||\, s[4i+1] \,||\, s[4i] \text{ for } 1 \leq i \leq 0 \quad (4)$$

2.2. Encryption and Decryption Algorithm Routine

The proposed cipher uses the Feistel network structure together with an S-Box and P-Box. It uses only 8 rounds to encrypt a 64-bit block data using a 128-bit key. In the Feistel structure of the cipher, each round consists of two stages. In stage 1, the input data passes through four unique processes and each process transforms the data. The processes include: Add_Round_Key layer, S-Box layer, P-Box layer and a second Add_Kound_Key layer. Stage 1 uses the first half of the 128-bit key. In stage 2, data from stage 1 is passed through the same processes as in stage 1, but the second half of the 128-bit key is used. The beauty about the Feistel structure is that encryption and decryption routine are the same with the keys in reverse order. Figure 1 shows the top-level algorithm description of the encryption and decryption routines.



(a) (b)

Figure 1. (a) Encryption Algorithm Description (b) Decryption Algorithm Description

2.3. Add_Round_Key Layer

Each stage for each round in the algorithm routine consists of two Add_Round_Key processes. The 128-bit key key[127:0] is divided into four registers, each of which is allocated 32-bit (K[0], K[1], K[2], K[3]) as shown in Equation 1. The Add_Round_Key is a simple XOR operation and is applied four times in each round. For the encryption routine, stage 1 in each round uses K[0] and K[1] while stage 2 in each round uses K[2] and K[3]. For the decryption routine, stage 1 in each round uses K[2] and K[3] while stage 2 in each round uses K[0] and K[1]. Considering the first Add_Round_Key of the encryption routine, let K[0] = key[31:0] and let the current state S[0] = state[31:0]. The output of the Add_Round_Key is therefore given as:

$$state[i] = key[i] \; XOR \; state[i] \; for \; 31 \leq i \leq 0 \tag{5}$$

This is the same for all Add_Round_Key layers for both encryption and decryption routines of the proposed cipher.

2.4. Substitution Box (S-Box) Layer

Claude Shannon in his 1945 classified report "A Mathematical Theory of Cryptography" [8] identified confusion and diffusion as two important properties of a secure cipher. Confusion is a process that drastically changes data from input to output. This is achieved in modern cryptography by translating data through a non-linear table called the S-Box. The eight S-Boxes used in the Data Encryption Standard (DES) [9] were studied intensely because experts were sure there was a backdoor into the algorithm, which latter proved to be false. To clear all doubts about a backdoor into our cipher, the decision was made to employ the S-Box of PRESENT algorithm. The PRESENT algorithm S-Box is resistant to both linear and differential cryptanalysis, and is also extremely lightweight. Table 2 shows the S-Box used in our proposed cipher. The S-Box is a 4-bit input to 4-bit output S-Box. For the S-Box layer, the current state S[31:0] is considered as 4-bit words W[0], W[1], …, W[7] where:

$$W[i] = S[4i+3] \,||\, S[4i+2] \,||\, S[4i+1] \,||\, S[4i] \text{ for } 7 \leq i \leq 0 \quad (6)$$

The output nibble S[W] provides the updated state. The encryption and decryption algorithms use the same S-Box.

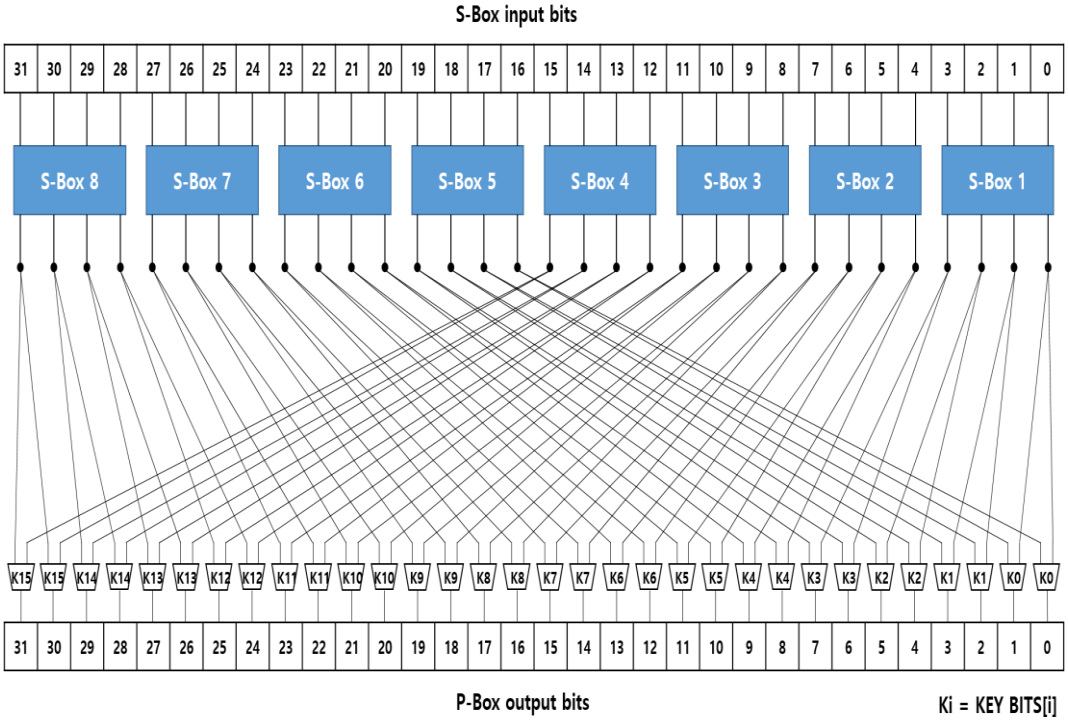
Table 2. Substitution Box (S-Box)

W	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S[W]	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

2.5. Permutation Box (P-Box) Layer

The diffusion aspect of Shannon's theory says that changing a single bit of the input should lead to many bit changes in the output. When done properly, every part of the input affects every other part of the output, thereby making cryptanalysis much more difficult. In modern cryptography, Shannon's diffusion is achieved by employing a permutation box (P-Box). Many block ciphers employ the use of fixed P-Boxes. With the advent of linear and differential cryptanalysis, a fixed P-Box is no longer secure. The P-Box proposed in this paper is the One Stage Omega Permutation Network P-Box. As shown in Figure 2, the P-Box consists of 32 2-to-1 multiplexors. A 16-bit variable KEY_BITS serve as the select signals to the multiplexors. The lower half of the 128-bit key (key[63:0]) is used to calculate the KEY_BITS as follows:

$$KEY\_BITS[15:0] = key[63:48] \wedge key[47:32] \wedge key[31:16] \wedge key[15:0] \quad (7)$$



(a)

```
j=0
For ( i=0; i<16; i=i+1 ) {
    if (!KEY_BITS[i]) {              // Mux select signal – Key bits
        pbox[j] = in_data[i]         // 1 Key bit = 2 in_data → 2 pbox
        pbox[j+1] =in_data[i+16]     // 16 Key bit = 32 in_data → 2 pbox
    }
    else {
        pbox[j] = in_data[i+16]
        pbox[j+1] = in_data[i]
    }
    j=j+2
}
```

(b)

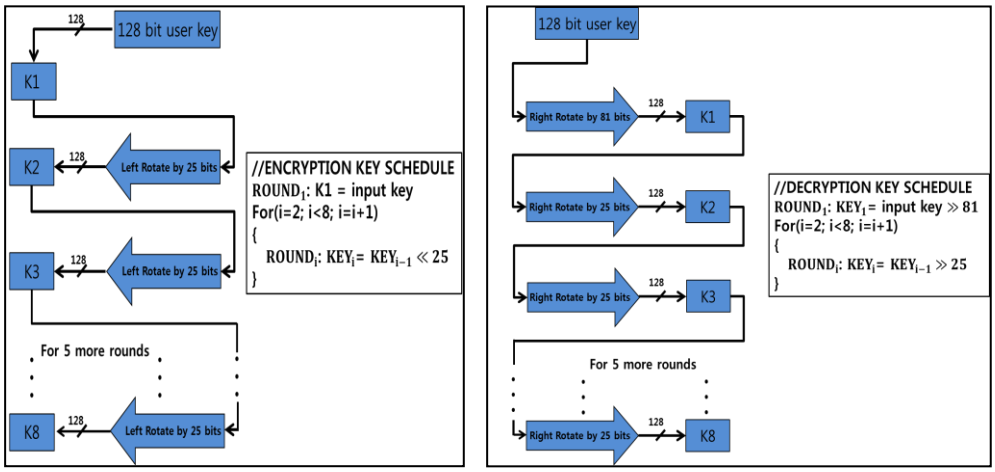
Figure 2. (a) Top Level Description of Permutation Box (P-Box) (b) P-Box Algorithm

The algorithm for generating the outputs of the P-Box is also shown in Figure 4. In the algorithm, in_data[31:0] indicates the outputs of the S-Box, and pbox[31:0] indicates the output of the P-Box. The encryption and decryption algorithms use the same P-Box.

2.6. Key Schedule Algorithm

The algorithm is designed with a 128-bit key, which is more than enough to prevent a brute force attack. For our cipher structure, the only difference between the encryption structure and the decryption structure is the key schedule algorithms. The key schedule algorithm is very simple since a lightweight cipher does not necessarily have to be as

strong as the standard algorithms. The key schedule algorithm for both encryption and decryption are shown in Figure 3.



(a) (b)

Figure 3. (a) Encryption Key Schedule (b) Decryption Key Schedule

3. Experimentation and Results

The proposed cipher was designed using Verilog HDL and was verified using FPGA. Xilinx Virtex6 XC6VLX70 was used for the purpose of synthesis, and Mentor Graphics Modelsim SE-64 10.1c was used for the purpose of simulation. Figure 4 shows simulation results of an encryption/decryption core of the cipher, and Table 3 shows the synthesis results of the proposed cipher compared with other ciphers.

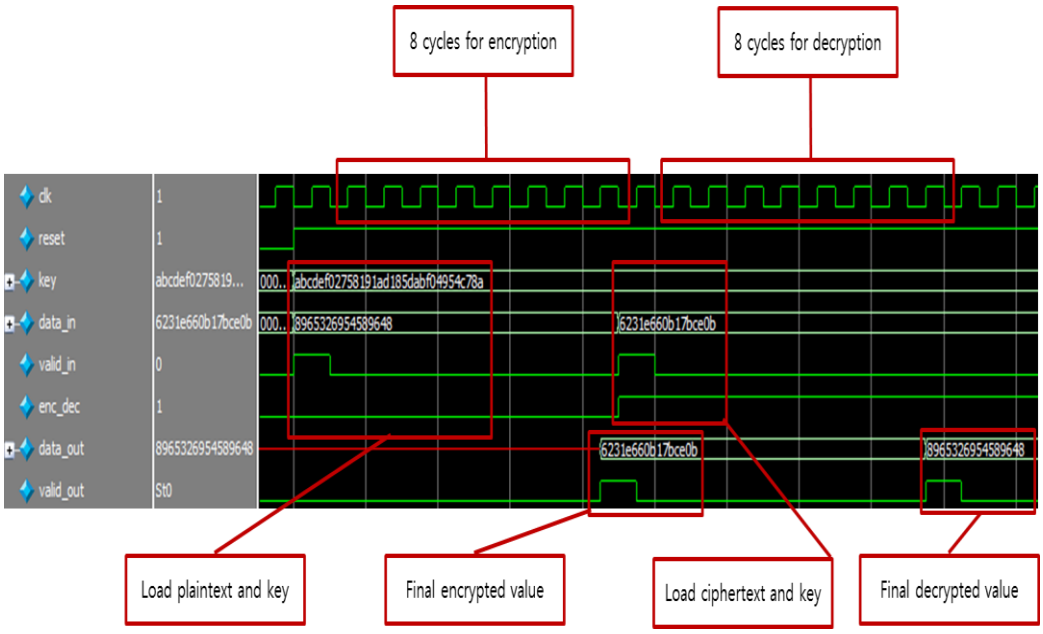


Figure 4. Encryption/Decryption Core Simulation

Table 3. Synthesized Results Comparison

Algorithm	Max delay (ns)	Cycles/block	Block size (bits)	Key size (bits)	Area (slices)	Throughput (Mbps)	Throughput/area (Mbps/slice)
PRESENT[10]	3.94	31	64	128	202	508	2.5
AES[11]	20.00	46	128	128	222	139	0.62
CLEFIA[12]	5.4	36	128	128	270	658	2.4
XTEA[11]	15.97	112	64	128	254	35.78	0.14
Proposed	2.97	8	64	128	196	2693	13.76

3.1. Image Encryption/Decryption Application

Here, a 480x272 JPEG format image is encrypted and decrypted using the proposed cipher. This is to investigate the real time processing performance of our design. The experiment is carried out on iNEXT-v6 test board. Here, the original image is converted into a coe file and stored in on-chip block ROM. Data from the ROM is sent to the encryption module and after encryption the data is stored in a RAM. The data from the RAM is sent to the decryption module and after decryption, another RAM is used to store this data. The original date, encrypted data, and decrypted data take turns to be displayed on the LCD of the iNEXT-v6 test board. This is shown in Figure 5. To investigate the time taken for both encryption and decryption of the image, the clock period of the simulation was set at 10ns and the time for the entire encryption and decryption was taken. The result of this experiment is shown in Table 4.

Table 4. Encryption/Decryption Application Comparison

Algorithm	Image size (Pixels)	Clock period (ns)	Cycles/block	Required time for ENC/DEC (S)
AES	480x272	10	46	0.120
PRESENT	480x272	10	32	0.080
TEA	480x272	10	64	0.160
Proposed	480x272	10	8	0.021

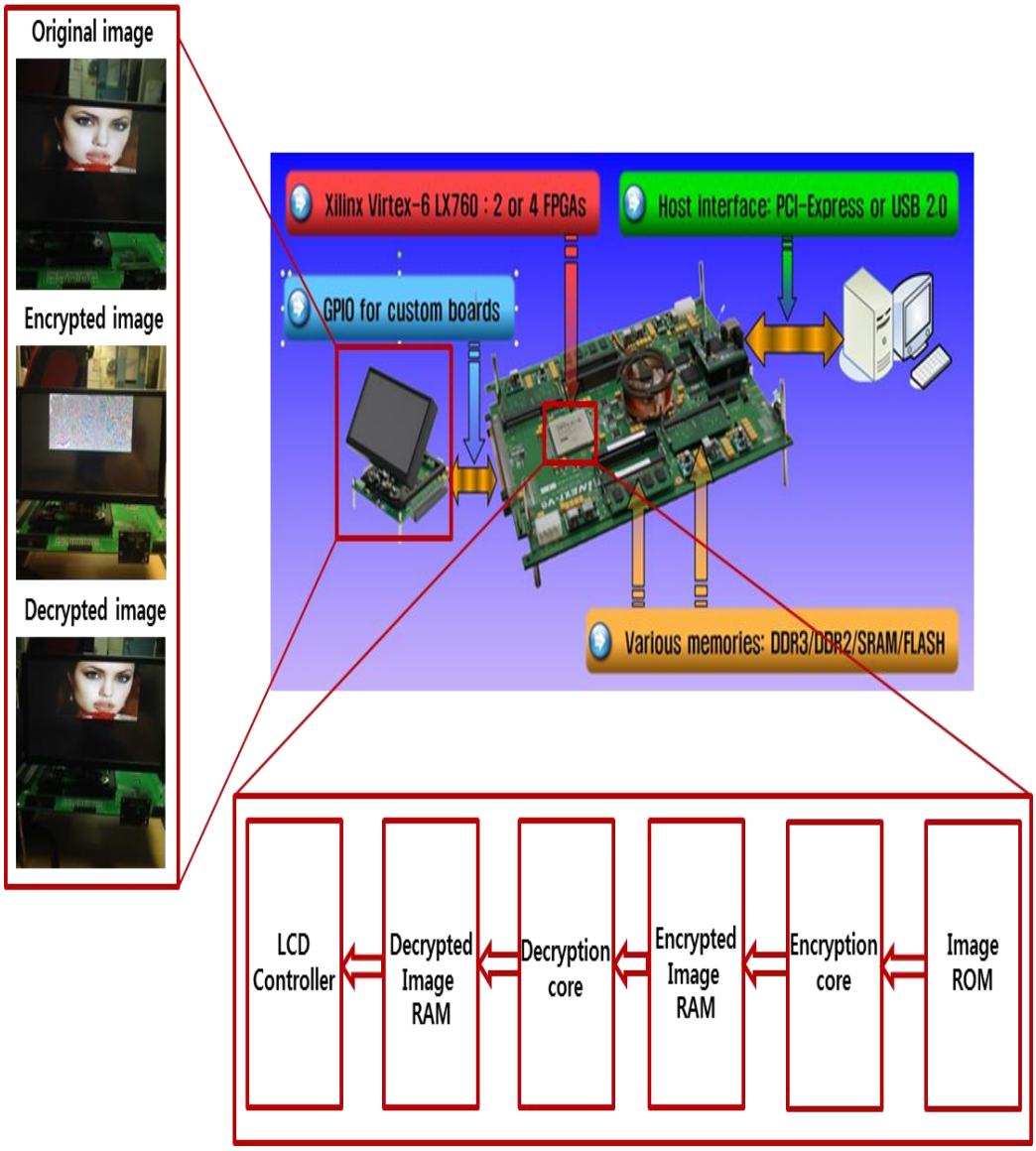


Figure 5. Image Encryption/Decryption Application

Appendix

Here, we present some test vectors for the proposed cipher. We keep the 128-bit cipher key (key [127:0] = ABCD_EF02_7581_91AD_185D_ABF0_4954_C78A) constant while changing the plaintext by 1-bit. This is to test the avalanche nature of the cipher. The hamming distance (bit change in two ciphertexts with plaintext change of 1-bit) is calculated by taking the exclusive-or of the two ciphertexts. All the data are expressed in hexadecimal notation and shown in Table 5.

Table 5. Investigating the Avalanche Effect

Plaintext	Ciphertext	Hamming distance
0000 0000 0000 0000	D0EB BFB0 02FC 211E	28
0000 0000 0000 0001	A801 3D57 25FC 8496	
0000 0000 0000 0002	1B9A 72BA CD39 8D34	29
0000 0000 0000 0003	D256 67B0 9B4D B802	
0000 0000 0000 0004	936F 3EDA 6019 7859	35
0000 0000 0000 0005	8959 DC89 9621 A7CF	
0000 0000 0000 0006	EB5B 8A40 466B AEC6	30
0000 0000 0000 0007	A884 59CA 673E 9FA2	
0000 0000 0000 0008	8906 477C 8E40 B4C7	24
0000 0000 0000 0009	45AF 2659 DDC0 9CF5	

Acknowledgment

This research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the Global IT Talent support program (IITP-2016-R0134-16-1019) and Human Resource Development Project for Brain scouting program(IITP-2016-R2418-16-0007) supervised by the IITP(Institute for Information and Communication Technology Promotion).

References

[1] M. Weiser, "The Computer for the 21st Century", Scientific America Issue on Communication, Computers and Networks, **(1991)** September.
[2] M. Feldhofer, S. Dominikus and J. Wolkerstorfer, "Strong Authentication for RFID Systems Using AES Algorithm", Cryptographic Hardware and Embedded Systems - CHESS, LNCS 3156, Springer-Verlag, **(2004)**, pp. 357-370.
[3] National Institute of Standards and Technology, "Advanced Encryption Standard (AES)", Federal Information Processing Standards Publication 197, **(2001)**.
[4] G. Bansod, N. Raval and N. Pisharoty, "Implementation of a New Lightweight Encryption Design for Embedded Security", IEEE Transaction on Information Forensics and Security, vol. 10, **(2015)**, pp. 142-151.
[5] A. Bogdanov, C. Paar and A. Poschmann, "PRESENT: An Ultra-Lightweight Block Cipher", Cryptographic Hardware and Embedded Systems - CHESS, LNCS 4727, Springer-Verlag, **(2007)**, pp. 450-466.
[6] W. L. Cho, K. B. Kim and K. W. Shin, "A Hardware Design of Ultra-Lightweight Block Cipher Algorithm PRESENT for IoT Applications", Journal of Information and Communication Convergence Engineering, vol. 20, No. 7, **(2016)** July, pp. 1296~1302.
[7] G. C. Bae and K. W. Shin, "An Efficient Hardware Implementation of Lightweight Block Cipher Algorithm CLEFIA for IoT Security Applications", Journal of Information and Communication Convergence Engineering, vol. 20, No. 2, **(2016)** February, pp. 351~358.
[8] C. E. Shannon, "A Mathematical Theory of Cryptography", Bell System Memo, MM 45-110-02, **(1945)**.
[9] National Institute of Standards and Technology, "Data Encryption Standard (DES)", Federal Information Processing Standards Publication 197, **(2001)**.
[10] M. Sbeiti, M. Silbermann, A. Poschmann and C. Paar, "Design Space Exploration of PRESENT Implementation for FPGA", 5th Southern Conference on Programmable Logic, Sao Paulo, Brazil, **(2009)** April 01-03.
[11] Y. Panasayya and K. Jen-Peter, "Lightweight Cryptography for FPGAs", International Conference on Reconfigurable Computing and FPGAs, Quintana Roo, **(2009)** December 09-11.
[12] P. Paulo and C. Ricardo, "Compact CLEFIA Implementation on FPGAs", International Conference on Field Programmable Logic and Applications, Chania, **(2011)** September 05-07.

Authors

Gookyi Dennis Agyemanh Nana received his BSC Degree in Computer Engineering from the Kwame Nkrumah University of Science and Technology, Ghana in 2013. He is currently pursuing a MENG Degree in Information and Communication Engineering at Hanbat National University, South Korea. His research interests include SoC Design and Verification Platforms and Lightweight Cryptography.

Seungyong Park received his BS Degree and MENG Degree in Information and Communication Engineering from Hanbat National University, South Korea, in 2010 and 2012 respectively. He is currently pursuing a PhD Degree in Information and Communication Engineering at Hanbat National University, South Korea. His research interests include SoC Design and Verification Platforms, Image Signal Processing and Multimedia Codec Design.

Kwangki Ryoo received his BS, MS and PhD Degrees in Electronic Engineering from Hanyang University, South Korea in 1986, 1988 and 2000 respectively. From 1991 to 1994, he was an Assistant Professor at the Military Academy in South Korea. From 2000 to 2002, he worked as a Senior Researcher at ETRI, South Korea. From 2010 to 2011, he was a Visiting Professor at University of Texas at Dallas. Since 2003, he has been a Professor at Hanbat National University, South Korea. His research interests include Engineering Education, SoC Design and Verification Platforms, Image Signal Processing and Multimedia Codec Design.