FOREX Publication
Open Access | Rapid and quality publishing

International Journal of
Electrical and Electronics Research (IJEER)
Research Article | Volume 10, Issue 2 | Pages 245-249 | e-ISSN: 2347-470X

128-Bit LEA Block Encryption Architecture to Improve the Security of IoT Systems with Limited Resources and Area

Hyogeun An[1], Sudong Kang[2], Guard Kanda[3] and Kwangki Ryoo[4]
[1,2,3,4]Department of Information and Communication Engineering Hanbat National University, Daejeon, South Korea

***Correspondence:** Kwangki Ryoo; kkryoo@gmail.com; Tel.: (KR) +82-10-5234-0569

ABSTRACT- The LEA block encryption algorithm is an architecture suitable for IoT systems with limited resources and space. It was developed by the National Security Technology Research Institute in 2013 and established as an international standard for cryptography by the International Electrotechnical Commission in 2019, drawing much attention from developers. In this paper, the 128-bit LEA block encryption algorithm was light weighted and implemented in a hardware environment. All modules share and reuse registers and are designed and implemented in a bottom area through the resource sharing function. As a result of synthesis using Xilinx ISE 14.7 Virtex-5 as a design environment, the maximum frequency achieved 190.88 MHz and has a processing speed of up to 128 Mbps. Compared to the previously designed architecture, we present a bottom-level hardware design with a 128-bit LEA algorithm implemented with a 49.8% reduction in Flip-Flop, 18.8% reduction in LUTs, and 67.6% reduction in Slices.

Keywords: 128-bit LEA, IoT, Hardware Design, Lightweight, Cryptography Algorithm.

ARTICLE INFORMATION
Author(s): Hyogeun An, Sudong Kang, Guard Kanda and Prof. Kwangki Ryoo
Received: 11/04/2022; **Accepted**: 19/05/2022; **Published**: 10/06/2022;
e-ISSN: 2347-470X;
Paper Id: IJEER22TK405;
Citation: 10.37391/IJEER.100232
Webpage-link:
https://ijeer.forexjournal.co.in/archive/volume-10/ijeer-100232.html

Publisher's Note: FOREX Publication stays neutral with regard to Jurisdictional claims in Published maps and institutional affiliations.

1. INTRODUCTION

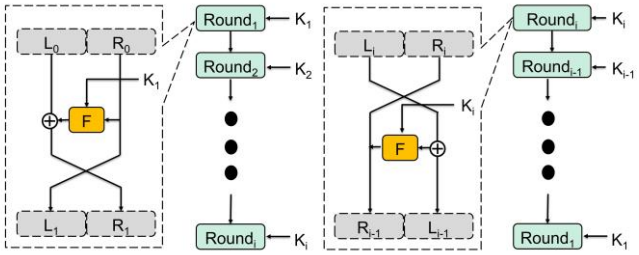
Data has become a vital commodity in the era of the Fourth Industrial revolution. As the Fourth Industrial Revolution broke out, most of the information and documents stored as data [1]. Accordingly, data exchange takes place online, and data has become an important resource [2,3]. Therefore, data has commercial value, such as being used for e-commerce, personal information, and cryptocurrency. Several encryption technologies are applied to protect this important data [4,5,16]. Furthermore, various encryption technologies are evolving or are securing developed. Currently, ubiquitous technologies such as IoT are applied in various fields such as healthcare, education, energy, and smart home [6,7,17]. As the domestic and industrial consumption of these technology increase, network attacks on IoT technologies are also increase [8]. To solve this problem, when applying existing SPN-structured cryptographic algorithms such as AES [9] or ARIA [10] in software, the code size increases, and problems such as large memory usage and decreased speed occur, and lightweight cryptographic algorithms such as HIGHT [11] and LEA [12] of ARX structure were developed. The algorithm realized weight reduction by applying the concept of the Feistel structure and applying Addition, Rotation, and XOR operations with low computational costs. To reduce the hardware area of the LEA algorithm architecture, an architecture using minimal operational function maintenance, register reuse, and resource sharing is proposed.

2. LEA ALGORITHM

2.1 Feistel Structure

The LEA algorithm is a block cipher algorithm based of the Feistel structure [13]. The Feistel structure is a type of block cipher that alternately performs substitution and permutation. Since the Feistel structure uses the same components for encryption and decryption, it is not necessary to configure different algorithms, so it is suitable for area reduction. The encryption method of the Feistel structure divides the information to be encrypted into two half's of information of the same bit length (L_0, R_0). A secret key used in each round is defined as K_i, and a function performed in each round is defined as F. In each round, the following operations are named R_0 of the first round is stored in L_1 of the next round, and R_0 is substituted into the round function F using the key K_1. The calculated value is added to the value of L_0 and then stored in R_1 of the next round. This is repeat until the last round. After all rounds are finished, L_i and R_i are finally encrypted half values. Conversely, in decryption, L_i and R_i are calculated in reverse order of encryption, and when all rounds are completed, L_0, R_0 becomes the decrypted half values. *Figure 1* shows the encryption/decryption flow of the Feistel structure. It is a process of encrypting from round 1 to round i and decrypting from round i to round 0 in reverse order. LEA cipher security is further strengthened by applying the GFN-Type3 structure that expanded the above Feistel structure.

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Electrical and Electronics Research (IJEER)
Research Article | Volume 10, Issue 2 | Pages 245-249 | e-ISSN: 2347-470X



(a) Encryption (b) Decryption

Figure 1: Feistel-based Structure

2.2 128-bit LEA Algorithm Enc/Dec Process

The 128-bit standard LEA encryption process consists of a key schedule function that generates 24 round keys for 192-bit encryption with a 128-bit secret key and an encryption function that converts 128-bit plaintext into 128-bit ciphertexts using round keys and round functions. *Figure 2* shows the encryption/decryption architecture of LEA. The encryption/decryption round key $RK_{enc/dec}$ is generated through the key schedule function with the secret key, and $Round_{enc/dec}$ is performed with the generated round key and plaintext/ciphertexts.

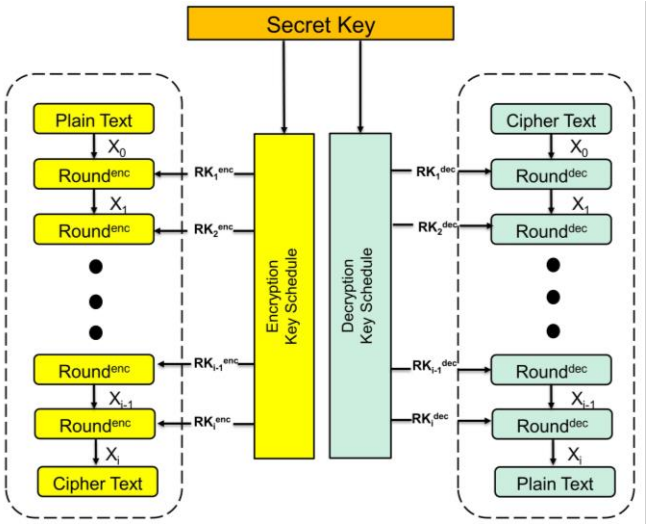


Figure 2: Architecture of Encryption/Decryption

3. PROPOSED LEA 128-BIT BLOCK ENCRYPTION ARCHITECTURE

3.1 Top Module

The top module is encapsulating module of the 128-bit sub module. It has a round key generator, an encryption/decryption module, a module controller, two 128-bit registers, and one 32-bit register. The length of key required for the round function of the 128-bit LEA algorithm is 192-bit. However, 128-bit round keys are generated, and the remaining 64-bit is rearranged and used. Therefore, the actual required register size is 128-bit. The two 128-bit registers in the top module are used as round key buffers and buffers that store a single round of encryption/decryption data, respectively. One 32-bit register is used to store the previous data required in the encryption/decryption function. The select signal of the top module functions to choose between encryption/decryption mode of operation. In the top module, encryption/decryption is performed by inputting plaintext or ciphertext. Two counters are used within the top module. The first counter divides 128-bit into 32-bit units and uses them to determine the bit address of the divided round key and ciphertext to process. The second counter is used to control the repetition of the round function from the first round to the last round during encryption/decryption. *Figure 3* shows the internal architecture and flow of data in which 128-bit plaintext blocks are calculated in 32-bit units using a round key module, an encryption/decryption module, and a counter. The top module was able to recycle registers to generate round keys that required 192-bit using only 128-bit of registers. As a result, the area of the entire module is reduced. Round data is controlled in 32-bit units, and data is rearranged in 8-bit units in the first and last rounds.

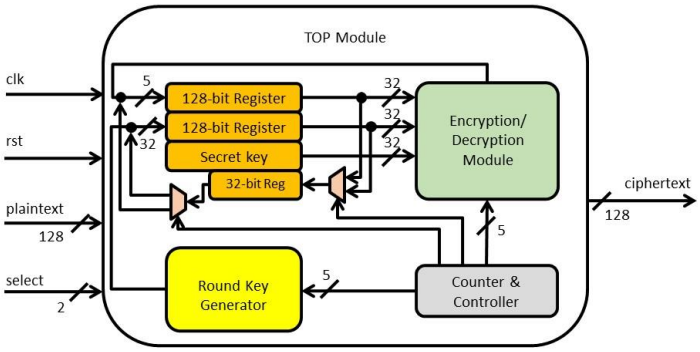


Figure 3: Proposed Overall LEA Architecture

3.2 Round Key Generator Module

One round key is generated by dividing 128-bit round keys into 32-bit (T[x]) through ARX operations during four clock cycles.

The LEA algorithm uses the same master round key from the first round to the 24th round for both encryption and decryption. During encryption, 24 round keys are generated in the encryption/decryption module using the expression in *Table 1*, sequentially from the first round key to the 24th round key. During decryption, 24 round keys are generated in the encryption/decryption module using the expression in *Table 2* in reverse order from the 24th round key to the first round key.

Table 1. LEA Encryption Key Generation Algorithm

Bit place	Formula
T[0]	$ROL_1(T[0] \boxplus ROL_i([i \bmod 4]))$
T[1]	$ROL_3(T[1] \boxplus ROL_{i+1}([i \bmod 4]))$
T[2]	$ROL_6(T[2] \boxplus ROL_{i+2}([i \bmod 4]))$
T[3]	$ROL_{11}(T[3] \boxplus ROL_{i+3}([i \bmod 4]))$

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Electrical and Electronics Research (IJEER)
Research Article | Volume 10, Issue 2 | Pages 245-249 | e-ISSN: 2347-470X

Table 2. LEA Decryption Key Generation Algorithm

Bit place	Formula
T[0]	$ROR_i(T[0]) \boxminus ROL_i([i \bmod 4]))$
T[1]	$ROR_3(T[1]) \boxminus ROL_{i+1}([i \bmod 4]))$
T[2]	$ROR_6(T[2]) \boxminus ROL_{i+2}([i \bmod 4]))$
T[3]	$ROR_{11}(T[3]) \boxminus ROL_{i+3}([i \bmod 4]))$

Figure 4 implements the expressions described in *Tables 1* and *2* in hardware, and the 32-bit variables stored in the register are selected according to round, and the selected variables are generated 32-bit by calculating ROL/ROR, add, subtract, and ROL/ROR according to encryption or decryption. The round key generation module was implemented in a bottom area using resource sharing, and the processing speed during decoding was optimized by calculating the round key generation function in reverse order.

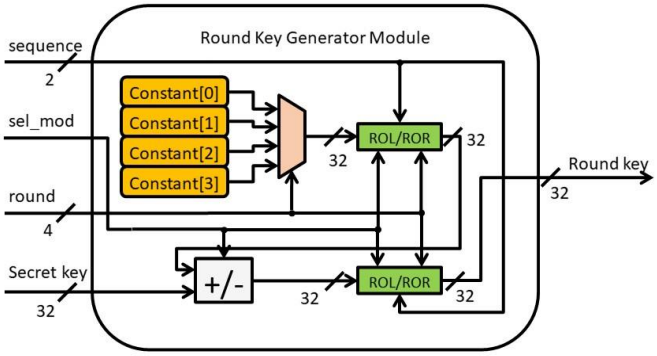


Figure 4: Proposed Architecture of LEA Round Key Generator

3.3 Encryption/Decryption Module

The encryption/decryption module has functions that perform encryption and decryption, and encryption or decryption is determined according to the select signal. For each round, the encryption/decryption module operates three times, and a total of three clock cycles are required to encrypt 128-bit of data. The encryption module performs encryption by receiving a round key ($RK_i^{enc/dec}$) divided into 32-bit and data under encryption (X_i) divided into 32-bit. In the encryption process, as shown in *Table 3*, two plaintext blocks divided into 32-bit and two round key blocks divided into 32-bit are generated through ARX operations

Table 3. LEA Encryption Function Algorithm

Bit place	Formula
$X_{i+1}[0]$	$ROL_9((X_i[0] \oplus RK_i^{enc}[0]) \boxplus (X_i[1] \oplus RK_i^{enc}[1]))$
$X_{i+1}[1]$	$ROL_5((X_i[1] \oplus RK_i^{enc}[2]) \boxplus (X_i[2] \oplus RK_i^{enc}[3]))$
$X_{i+1}[2]$	$ROL_3((X_i[2] \oplus RK_i^{enc}[4]) \boxplus (X_i[3] \oplus RK_i^{enc}[5]))$
$X_{i+1}[3]$	$X_i[0]$

Figure 5 shows an encryption process of LEA in which the encryption process of the i^{th} round is sequentially performed three times, and the previously calculated $X_i[0]$ value is stored in a 32-bit buffer in the third cycle, and $X_{i+1}[0]$, $X_{i+1}[1]$, $X_{i+1}[2]$ and $X_{i+1}[3]$ values of the next round are stored.

In the decoding process, as shown in *Table 4*, two 32-bit ciphertext blocks and two 32-bit round key blocks are generated through ARX operations. *Figure 6* shows a decryption process of LEA in which the decryption process of the i-th round is sequentially performed three times, and the previously calculated $X_i[3]$ value is stored in a 32-bit buffer in the third cycle and $X_{i+1}[0]$, $X_{i+1}[1]$, $X_{i+1}[2]$ and $X_{i+1}[3]$ values of the next round are stored.

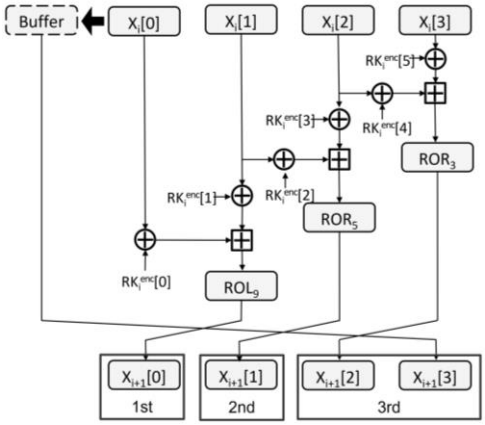


Figure 5: Algorithm of the Proposed LEA Encryption Module

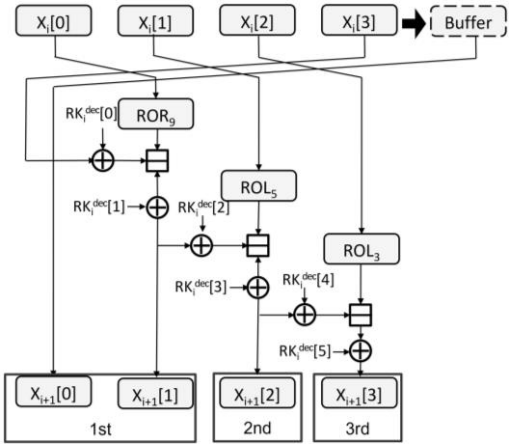


Figure 6: Algorithm of the Proposed LEA Decryption Module

Table 4. LEA Decryption Function Algorithm

Bit place	Formula
$X_{i+1}[0]$	$X_i[3]$
$X_{i+1}[1]$	$(ROR_9(X_i[0]) \boxminus (X_{i+1}[0] \oplus RK_i^{dec}[0])) \oplus RK_i^{dec}[1]$
$X_{i+1}[2]$	$(ROR_5(X_i[1]) \boxminus (X_{i+1}[1] \oplus RK_i^{dec}[2])) \oplus RK_i^{dec}[3]$
$X_{i+1}[3]$	$(ROR_3(X_i[2]) \boxminus (X_{i+1}[2] \oplus RK_i^{dec}[4])) \oplus RK_i^{dec}[5]$

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Electrical and Electronics Research (IJEER)
Research Article | Volume 10, Issue 2 | Pages 245-249 | e-ISSN: 2347-470X

Figure 7 shows the internal architecture of the encryption/decryption module that performs the encryption/decryption process shown in *Figure 5 and 6* and the flow of data accordingly. The encryption/decryption module performs encryption/decryption by receiving two 32-bit plaintext/ciphertext and two 32-bit current round keys from the top module.

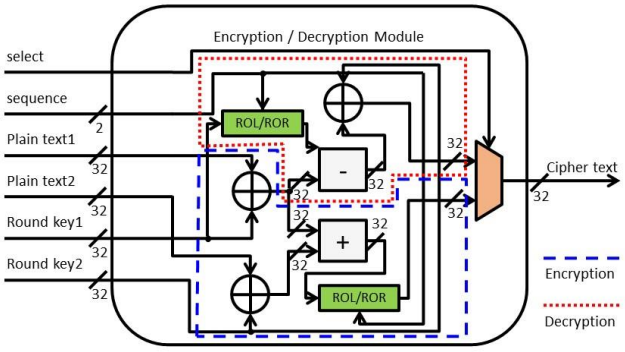


Figure 7: Combined of Encryption/Decryption Module

4. VERIFICATION AND ANALYSIS OF PROPOSED ARCHITECTURE

The proposed ultra-lightweight LEA 128-bit LEA encryption/decryption module was designed in Verilog HDL using Xilinx ISE 14.7 and tested and implemented on a virtex-5 FPGA device. To confirm the correct operation, *Figure 8* shows the simulation results of the designed LEA module. When 128-bit plaintext *"128'h10111213_14151617_18191a1b_1c1d1e1f"* and 128-bit secret key *"128'h0f1e2d3c_4b5a6978_8796a5b4 c3d2e1f0"* were input, 128-bit ciphertext *"128'h9fc84e35_28c6c618_5532c7a7_04648bfd"* was output. Conversely, when the 128-bit ciphertext *"128'h9fc84e35_28c6c618_5532c7a7_04648bfd"* was decrypted using the 128-bit secret key *"128'h0f1e2d3c_4b5a6978_8796a5b4 c3d2e1f0"*, it was confirmed that the 128-bit plaintext *"128'h10111213_14151617_18191a1b_1c1d1e1f"* was output, such as the plaintext before encryption.

The maximum operating frequency of the proposed ultra-lightweight LEA encryption/decryption architecture is 190.88 MHz, requires 191 clock cycles during encryption as shown in *Table 5*, and has a throughput of 128 Mbps. A 283 clock cycle is required for decryption and has a throughput of 86 Mbps. The encryption-only architecture of the ultra- lightweight LEA module occupied or required 296 FFs, 501 LUTs, and 363 slices as shown in *Table 5*, and the encryption/decryption architecture of the ultra-lightweight LEA module occupied 301 FFs, 1151 LUTs, and 485 slices. Compared to the data analyzed in papers using Virtex-5 Xilinx FPGA, which is most similar to the verification environment of the proposed design, FF decreased by about 46% compared to the synthesis results of Yoon et al. [14] as shown in *Table 5*. LUTs decreased by about 18%, slice decreased by about 53%, and Sung et al. [15] module results.

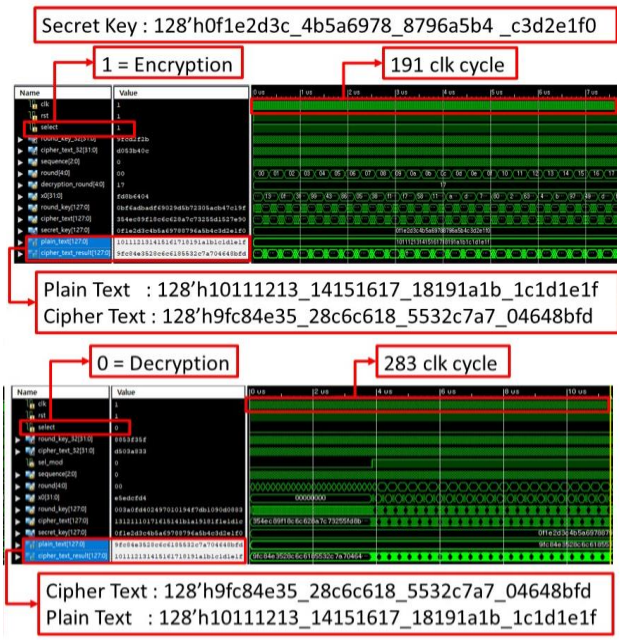


Figure 8: Simulation Result of Proposed LEA Architecture

Table 5. Synthesis Result of 128-bit LEA Module Compared to Some Existing Designs

Cipher Core Architecture	FF	Result	LUTs	Result	Slices	Result
Encryption Core	296	46.4% Reduced	501	18.7% Reduced	363	53.2% Reduced
[14]	552		616		775	
Enc/Dec Core	301	49.8% Reduced	1151	18.8% Reduced	485	67.6% Reduced
[15]	600		1418		1498	

5. CONCLUSION

Lightweight cryptographic technologies related to IoT are rapidly emerging and evolving particularly in the era of the fourth industrial revolution. Because IoT environments use low power and low budget, computing performance and area cannot be secured. To address this issue, we propose a hardware design of the ultra-lightweight 128-bit LEA algorithm by further reducing its size based on the LEA algorithm that features low area implementation through the use of low power and simple architecture operations. It was verified that the 128-bit LEA block encryption/decryption cipher architecture was implemented with a smaller area than the existing proposed LEA encryption/decryption module by reusing the internal register of the module and sharing resources to make it lightweight. The proposed block encryption/decryption module is capable of both encryption and decryption and is designed to be ultra-lightweight and

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Electrical and Electronics Research (IJEER)
Research Article | Volume 10, Issue 2 | Pages 245-249 | e-ISSN: 2347-470X

suitable for IoT systems that require smaller sizes, low power, and low cost. In the future, we will modify the LEA algorithm to support 32-bit and 64-bit block encryption/decryption and further conduct research on SoC design and verification of encryption/decryption modules automatically using a synthesizable hardware processor

REFERENCES

[1] Kishor M. D. 2017. Paperless Society in Digital Era. International Journal of Library and Information Studies, 7(4), 317-319.

[2] Peterson, Z. N., Gondree, M., and Beverly, R. 2011. A position paper on data sovereignty: The importance of geolocating data in the cloud. In 3rd USENIX Workshop on Hot Topics in Cloud Computing.

[3] Ilmudeen A. 2020. Big Data, Artificial Intelligence, and the Internet of Things in Cross-Border E-Commerce. Cross-Border E-Commerce Marketing and Management. DOI:10.4018/978-1-7998-5823-2.ch011

[4] Mohan, D., Alwin, L., Neeraja, P., Deepak Lawrence K. and Vinod Pathari. 2021. A private Ethereum blockchain implementation for secure data handling in Internet of Medical Things. Journal of Reliable Intelligent Environments. DOI:10.1007/s40860-021-00153-2

[5] Dijesh P., SuvanamSasidhar B. and Yellepeddi V. 2020. Enhancement of e-commerce security through asymmetric key algorithm. Computer Communications, 8, 125-134. DOI:10.1016/j.comcom.2020.01.033

[6] Ryoo H. G. and Ryoo K. K. 2019. Kindergarten school bus notification service using IoT network. Journal of Next-generation Convergence Technology Association, 3(1), 21-28. DOI:10.33097/JNCTA.2019.03.01.21

[7] Akbar M. A., Rashid M. M. and Embong A. H. 2018. Technology Based Learning System in Internet of Things (IoT) Education. 2018 7th International Conference on Computer and Communication Engineering, 192-197. DOI: 10.1109/ICCCE.2018.8539334

[8] Tara S. 2021. IoT Attacks Skyrocket, Doubling in 6 Months. Threatpost. https://threatpost.com/iot-attacks-doubling/169224/

[9] Daemen J. and Rijmen V. 1998. AES Proposal : Rijndael. https://www.cs.miami.edu/home/burt/learning/Csc688.012/rijndael/rijndael_doc_V2.pdf

[10] Kwon D. et al. 2003. New Block Cipher: ARIA. Information Security and Cryptology, 2971, 432-445. DOI:10.1007/978-3-540-24691-6_32

[11] Hong D. et al. 2006. HIGHT: A New Block Cipher Suitable for Low-Resource Device. Cryptographic Hardware and Embedded Systems, 4249, 46-59. DOI:10.1007/11894063_4

[12] Hong D. J., Lee J. K., Kim D. C., Kwon D. S., Ryu K, H. and Lee D. G. 2014. LEA: A 128-Bit Block Cipher for Fast Encryption on Common Processors. The 14th International Workshop on Information Security Applications, 8267, 3-27. DOI : 10.1007/978-3-319-05149-9_1

[13] Feistel Block Cipher. 2021. tutorialspoint. https://www.tutorialspoint.com/cryptography/feistel_block_cipher.htm

[14] Yoon K. H. and Park S. M. 2015. A Study on Hardware Implementation of 128-bit LEA Encryption Block. Smart Media Journal, 4(4), 39-46.

[15] Sung M. J., and Shin K. W. 2015. A Small-area Hardware Design of 128-bit Lightweight Encryption Algorithm LEA. Journal of the Korea Institute of Information and Communication Engineering, 19(4), 888–894. DOI : 10.6109/JKIICE.2015.19.4.888

[16] Lee, S. Y. (2020). Blockchain for Medical Information of Personal Health Record System. Journal of Smart Technology Applications, 1(1), 13-20. DOI:10.21742/JSTA.2020.1.1.03

[17] Falah, A. A. (2021). Improving Learning Performance in Neural Networks. International Journal of Hybrid Innovation Technologies, 1(2), 27-42. DOI:10.21742/IJHIT.2021.1.2.02