İSTANBUL UNIVERSITY
CERRAHPAŞA

RESEARCH ARTICLE
Electrica 2020; 20(1): 71-80

LWE: An Energy-Efficient Lightweight Encryption Algorithm for Medical Sensors and IoT Devices

Sezer Toprak[1], Akhan Akbulut[2], Muhammet Ali Aydın[3], Abdül Haim Zaim[4]

[1]Department of Computer Engineering, Yıldız Technical University, İstanbul, Turkey
[2]Department of Computer Engineering, İstanbul Kültür University, İstanbul, Turkey
[3]Department of Computer Engineering, İstanbul University-Cerrahpaşa, İstanbul, Turkey
[4]Department of Computer Engineering, İstanbul Commerce University, İstanbul, Turkey

Cite this article as: Toprak S, Akbulut A, Aydın MA, Zaim AH. LWE: An Energy-Efficient Lightweight Encryption Algorithm for Medical Sensors and IoT Devices. *Electrica*, 2020; 20(1): 71-80.

ABSTRACT

In today's world, systems generate and exchange digital data frequently and face a much broader range of threats than in the past. Within the context of this unsafe ecosystem, it is crucial to protect the data in a quick and secure way. In this paper, it is proposed that a lightweight block cipher algorithm called LWE in the purpose of having an encryption algorithm that is light enough for restricted/limited hardware environments and secure enough to endure primal cryptanalysis attacks. The length of blocks to be encrypted is set to 64 bits and the key length is defined as 64 bits. It is targeted for IoT systems with low-end microcontrollers and body sensor area devices. The performance and security aspects of LWE are evaluated with well-known algorithms and it is observed that LWE can establish a basic security baseline for transmitting raw data without creating a heavy load on the network infrastructure.

Keywords: Energy-efficient, lightweight, cryptography, security, IoT

Corresponding Author:
Akhan Akbulut

E-mail:
a.akbulut@iku.edu.tr

Received: 21.11.2019

Accepted: 22.12.2019

DOI: 10.5152/electrica.2020.19082

Content of this journal is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.

Introduction

The growing demand for intelligence-enabled environments has triggered lighter, smaller and efficient use of computational power over devices. Such intelligence-enabled things referred as Internet of Things (IoT) [1] uses the Internet to offer information transfer, analysis, appliances and communication services using existing standards [2]. This includes data from many devices and analytical calculations so that the most valuable data can be shared with applications. The problem of data integrity, confidentiality and authenticity, which arises in the security and data privacy, should be taken into consideration [3].

Restricted computational and energy resources do not allow implementation of existing security mechanisms that have high time and space complexity. Taking these as motivation, applications for restricted devices are unlikely to require large amounts of data to be encrypted, and therefore there is no requirement for lightweight ciphers to have high throughput.

Moreover, attackers are lacking in data and computing capabilities in this cryptographic environment, which means that lightweight ciphers only need to acquire moderate security.

The primary consideration for lightweight ciphers is therefore hardware performance. In many ways, hardware efficiency can be measured: the critical path length, latency, clock cycles, power use, throughput, area requirements, etc. The most important parameter is the area requirement, since the small area requirements can efficiently minimize both costs and power [4].

Related Work

In recent years, lightweight encryption algorithms have been discussed many times to achieve the ultimate goal using different methods. The algorithms used for security do not always provide the exact security-efficiency balance. As the common approach of using block encryption

method performs significantly better in terms of security-efficiency between stream encryption method and hashing algorithms [5].

LBlock, which has a block length of 64-bit and a key length of 80-bit, uses a Feistel variant structure in its design. In addition, the round function uses the SP-network with 4 to 4 S-box confusion layers and a 4-bit permutation-diffusion layer. LBlock can provide sufficient security against known cryptanalysis, linear cryptanalysis, impossible differential cryptanalysis attacks [4].

PRESENT is a SP network example consisting of 31 rounds. The 64 bits block is supported and two 80-and 128bit key lengths. Each of the 31 rounds comprises an XOR operation for introducing a round key K, a linear bit by bit permutation and a non-linear replacement layer. A single 4bit S-box S is used in a non-linear layer and used 16 times in parallel in each round. Invasive hardware and side channel attacks are likely to be a threat, as they are for all primitive cryptographers [6].

Simon is a family of lightweight block ciphers published in June 2013, by the NSA. Simon was optimized for hardware implementation [7]. A Feistel cipher with a n-bit word, the Simon Block cipher is the length of the block 2n. As of 2018, Simon of any variant has not been successfully attacked in full round. Differential cryptanalysis attacks are the bestpublished attacks against Simon.

In Speck, there are two words in one block, but the words may be 16, 24, 32, 48, or 64 bits. The Speck has an ARX-design. It has a modular addition to nonlinearity and uses XOR and linear mixing rotations. The ARX (Addition/Rotation/XOR) building is in fact the most efficient software algorithm. The Specks employs Feistellike structure. A counter to block slide attacks and rotational attacks is included in the Speck main scheme [8].

TEA is a Feistel that ensures non-linearity, adding and subtracting. A dual change leads to the repeated mixing of all bits of the key and data. The number of rounds before a single bit change is close to 32, so that sixteen cycles can suffice and we suggest a maximum of 32 bits. The key is set to 128 bits, which is sufficient to avoid the effect of simple search techniques. Instead of wasting computer power with byte or 4 bit operations, it uses a sequence of word operations [9].

KLEIN [10] is a 64-bit and variable key length-64(80 or 96-bit) family of block ciphers. KLEIN's structure is a typical SPN network. Although key keys are frequently changed, KLEIN is used to build block-coding based Hash functions and message authentication codes. The key schedule should also take into account a proper safety complexity to avoid possible associated weaknesses while balancing performance.

64-bit block length and 128-bit key length with HIGHT [11], suitable for low cost, low power and ultra light implementation. HIGHT has an iterative 32-round Feistel network. HIGHT has a prominent feature of simple operations like XOR, *mod*28^ add-on, and the left bit rotation. it is hardwaredriven. HIGHT consists of retaining the original master key value after all blanking keys and subkey generations are generated. Because of this, both encryption and decryption processes generate subtasks on the flies.

Proposed Approach

The proposed encryption algorithm (LWE) is designed to be efficient to use for IoT devices, but also to provide the required level of security and efficiency. The advantages and aspects of the methods used in other cryptographic algorithms have been examined and a study addressing the needs has been put forward. The well-known block encryption algorithms such as; AES [12], PRESENT, SAFER [13], Square [14] algorithms use the Substitution-Permutation (SP) network. The SPN network receives a block of text and key as input and applies several alternate "rounds" or "layers" of the insertion boxes (S-box) and permutation boxes (P-box) to generate the ciphertext block. In addition, these applications fulfill Shannon's confusion and substitution principles. The S-box and P-box convert the (lower) blocks of input bits to output bits. It is common for these transformations to be efficient in hardware such as XOR and bitwise rotation. The key is given in the form of round keys, usually derived from it, in each turn. Other well known Blowfish [15], DES [16], 3DES [17], Simon, TEA, RC6 [18] encryption algorithms use the Feistel network structure. The Feistel network is a versatile encryption that divides the current internal state of the plaintext into two parts and operates in only one part of each round of encryption or decryption. Between rounds, the left and right sides of the internal states change sides. The biggest benefit of the Feistel network is that the same structure can be used for encryption and decryption. As a result of the study, which is revealed by using these two approaches together, the aim is to use the good aspects of both approaches to achieve sufficient level of encryption power and low energy consumption as well as to consume less space in memory.

The LWE block encryption algorithm encrypts 64-bit blocks by symmetric encryption with a 64-bit key. In symmetric cryptography, cryptography consists of rounds and mathematical operations (XOR, XNOR, Bit Shift) are performed in addition to confusion and substitution operations in each round. Increasing the number of rounds used to make the encryption process more secure increases the security up to a point. However, since the increase in the number of rounds has a direct effect on the spent energy, our aim is to use as few rounds as possible [19]. Encryption and decryption operations will be performed by using three rounds in the proposed algorithm design. In each type of feistel network, the mixing and placing operations are performed by processing the data in 32 bit and 4 bit.

S-Box

8-bit S-box is used in the encryption algorithm. Algorithm AES and DES algorithms such as complex SPN network and does not have too many rounds, as well as for effective substitution operations 4 different 8-bit S-box (K-L-M-N SBox'es) used. In addition, the S-box structure used in the previous studies about [20], which has been tested for safety and has a specific stan-

dard structure. However, some changes have been made to the design of the S-Box to be successful in tests such as Avalance of Change.

Key Scheduling

The most important element of symmetric encryption algorithms is the key used in encryption and decryption. Because the confidentiality of the information is entirely dependent on this key, security-enhancing measures should be taken to prevent access by attackers. The algorithms that use the feistel structure aim to provide this confidentiality by using many rounds and different keys in each round. However, it is aimed to increase the encryption strength by using Substitution-Permutation processes in rounds. Although it is recommended that the key bit length be as high as possible so as not to be affected by brute-force attacks to obtain the key, this means that too much energy is consumed for algorithms that will work on limited hardware. Therefore, the use of a 64-bit key length that provides medium security will create $_{2}64_{-1}$ different key trial possibilities, preventing data contained within the encrypted text from being broken out of available time. The block uses an F-function that is influenced by the carefully designed Khazad block code [20]. Khazad is not a feistel algorithm and follows a broad trial strategy. The broad trial strategy consists of several linear and nonlinear transformations that provide complexity of dependence of the output bits to the input bits. Also seen in Figure 1, It is aimed to reduce the similarity of keys used in loops by using S-function in substitution operations like expansion blocks of Figure 2.

- In the first step, the 64-bit key is divided into 32-bit pieces and processed through S-Function
- The two 32-bit pieces decomposed into 16 pieces of 4 bit
- Distribute and mix using 32-bit S-Function. The $_{Kp}(_{I})$ expression is a 32-bit key as an explicit text. As a
- result of S-function, $_{Ks(i)}$ value is obtained.
  $K_s(i) = S(K_p(I))$

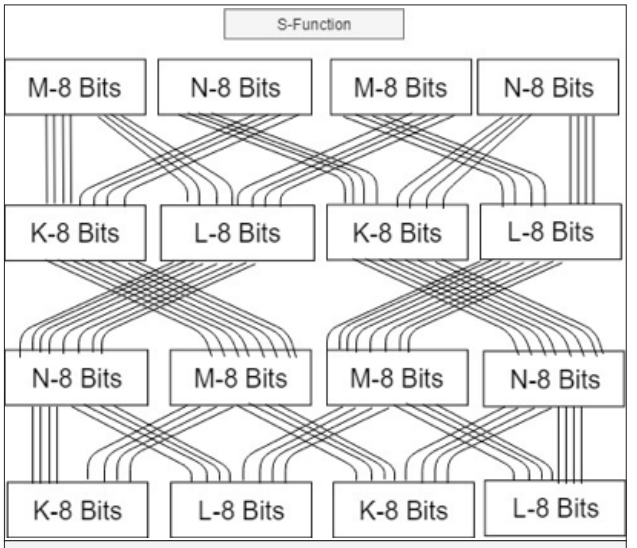


Figure 1. Inner structure of the S-function

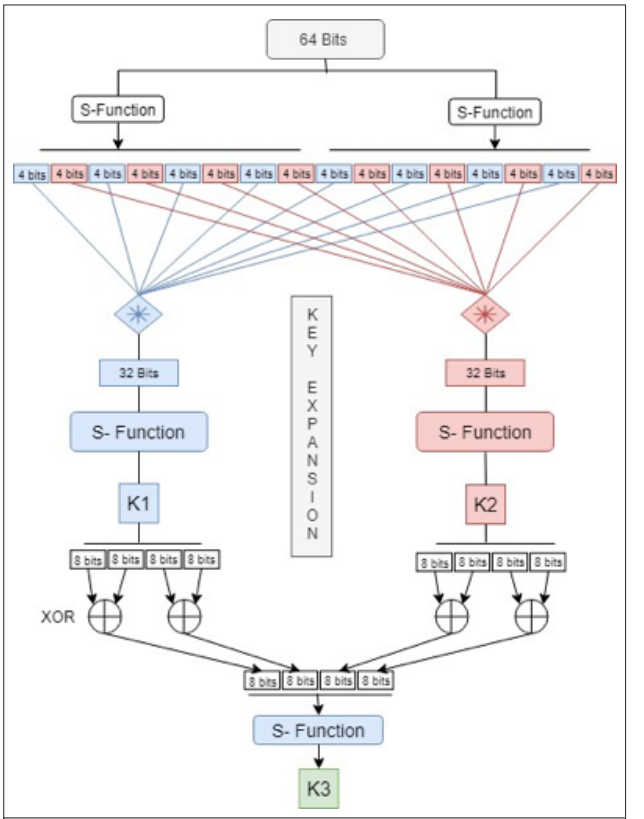


Figure 2. Inner structure of the key scheduling
Symbols:

M

→ XOR Operation

K

→ XNOR Operation

+ → Bit Concatenation

- S-function provides linear and non-linear changes using K, L, M and N tables.
- 4 different S-boxes are connected together in different ways to form a 32-bit S-Function.
- In this way, two different 32-bit round keys are generated using the 64-bit master key as a result of permutations and S-functions.
- 16 4-bit blocks are extracted as shown in Figure 2 to create 32-bit blocks.
- Last round key; XOR operation and permutations are performed with different 8-bit portions of the previously created round keys and the resulting 32-bit chain is passed through the S-Function.

Encryption

It is aimed to benefit from the features of both by using both Feistel network and SP (Substituion-Permutation) structure together. The keys obtained during the key generation stage were used in three different cycles within the Feistel network. It is aimed to reduce the relationship between keyencrypted text

by passing the bit chains obtained in the round through S-Functions. The procedure of the encryption operation is shown in Figure 3 and the encryption structure seen in Figure 4.

Decryption

In symmetric encryption algorithms, the decryption process is obtained by performing the reverse steps of the encryption process. Of course the S-Box structures used in encryption must be retrievable from the opposite. The procedure of the decryption operation is shown in Figure 5 and the decryption structure seen in Figure 6.

Evaluation

The purpose of the design is to provide a medium level of security and to work efficiently on limited hardware and to provide fast encryption. Although the LWE encryption algorithm has both Feistel network and SP network features, it contains the results of the existing security analyzes, but the evaluation of the changes in both performance and security are examined in detail under two headings.

1) Security Aspects*:* The S-Function used in the "Khazad legacy-level block cipher" study, which was reported to be successful against differential and linear attacks, has been shown to be successful against the same attacks.

Avalanche Test: Encryption algorithms are expected to be sensitive to changes made to the key. If the key is even one bit different from the original key, it means that the algorithm should not reveal the original data. Avalanche Test [21] is used to evaluate one bit of key text or the amount of changes that occur in the ciphertext by replacing plain text. If 50% of the bits change due to a one-bit change, the test is considered as valid.

$R^{1,1} = P_{0,0} \odot K1$

$R_{1'}2 = P_{0'}1 \oplus S(R_{1'}1)$ $R^{2,1} = R_{1,2} \odot K2$

$R_{2'}2 = R_{1'}1 \oplus S(R_{2'}1)$

$R^{3,1} = S(R_{2,2} \odot K3)$

$R^{3,2} = S(R_{2,1} \odot K3)$

Encrypted Text = $R_{3,2}$+$R_{3,1}$

Figure 3. Encryption Procedure

The following example was tested to measure the amount of bit changed in the encrypted text of the 1 bit value modified in the key used to encrypt.

It has been found that a 1-bit change in the encryption key causes a 33-bit change in the encrypted text. The average bit change was calculated by encrypting the bit differences that occur in the text encrypted using different encryption keys with 32 different keys and comparing the results. As a result, the bit change in the encrypted text using 32 different keys was found to be 33.28 bits on average. Looking at these test results, it is seen that the encryption algorithm passed the Avalanche Test.

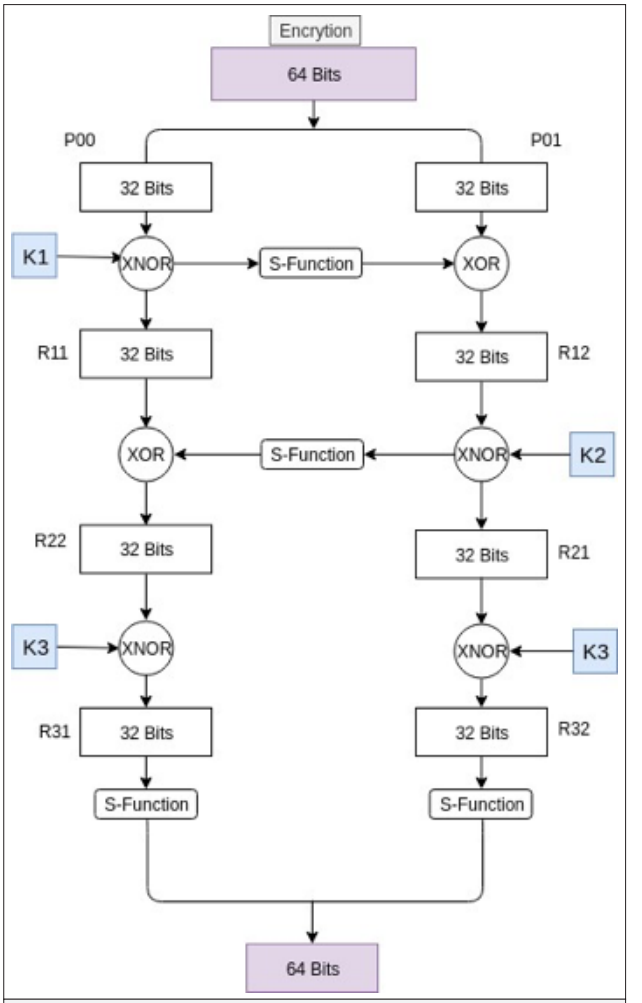


Figure 4. Encryption Structure

$R^{2,1} = S(R_{3,2}) \odot K3$

$R^{2,2} = S(R_{3,1}) \odot K3$

$R_{1'}1 = R_{2'}2 \oplus S(R_{2'}1)$

$R^{1,2} = R_{2,1} \odot K2$

$P^{0,0} = R_{1,1} \odot K1$

$R_{0'}1 = P_{1'}2 \oplus S(R_{1'}1)$

Decrypted Text = $P_{0,0}$+$P_{0,1}$

Figure 5. Encryption Structure

Linear Cryptanalysis: Linear cryptoanalysis attempts to exploit the high probability occurrences of linear expressions

including plain text bits and ciphertext bits and subkey bits. It is a known plaintext attack: that is, it is considered to have knowledge of cryptic texts corresponding to a series of plaintexts. However, there is no way for the attacker to choose which plain text (and the corresponding encrypted text) is available. In many applications and scenarios, it makes sense to assume that the attacker has a random set of plain text and corresponding cipher texts.

The basic idea is to approach the operation of a portion of the code with a linear expression, where linearity refers to a mod-2 operation (i.e, the specific-OR indicated by "XOR"). Such an expression is:

$$X_{i1} \oplus X_{i2} \oplus ... \oplus X_{iu} \oplus Y_{j1} \oplus Y_{j2} \oplus ... \oplus Y_{jv} = 0 \quad (1)$$

Here, *Xi* represents the *ith*-bit of *X*=[*X*1,*X*2,...] and *Yj* represents the $_j$*th*-bit of $_{Y=[Y1,Y2,....]}$. This equation represents summation the special-OR (⊕) of the input bits U and the output bits V.

The approach in linear cryptanalysis is to identify the above statements, which are likely to occur high or low. (The obvious linearity as above should not apply to all input and output values, otherwise the password will be negligently weak.) If an encryption algorithm is likely to occur as in equation (1) or shows a high probability of occurrence, it indicates that the password is of poor randomness.

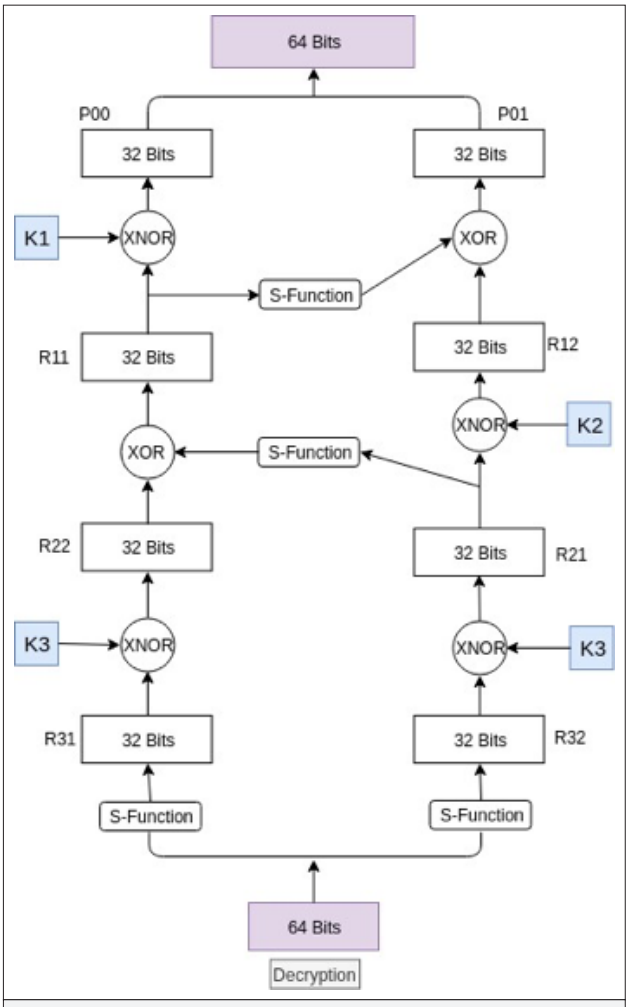


Figure 6. Decryption Structure

Table 1. An example of a avalance test sample-1

Name	Bit String
Plain Text	00110000001100000011000000110000
	00110000001100000011000000110001
Key	00110000001100000011000000110000
	00110000001100000011000000110000
Cipher Text	01010101001000011110100010001001
	01110000100010110101000111010000

Table 2. An example of a avalance test sample-2

Name	Bit String
Plain Text	00110000001100000011000000110000
	00110000001100000011000000110001
Key	00110000001100000011000000110000
	00110000001100000011000000110001
Cipher Text	01111000100001111011101101101111
	00010110111010111011101101000111

There are several ways to perform a linear cryptanalysis attack. In this attack, we will implement the algorithm proposed by Matsui [22]. The input bits in the Algorithm (1) are represented by X and the output bits by Y to indicate that there is a linear approach.

$$Pr(X_1 \oplus ... \oplus X_n = 0) = {}^{2}_ + 2^{n-1} \prod i=1 \varepsilon_i \quad (2)$$

If the sum of the respective subkey bits is "0", the bias of (1) will have the same sign (+ or -) as the bias of the expression containing the subkey sum, and the bias of 1, (1) has the opposite sign It will be.

When $_{PL}$ = 1, it means that the linear expression (1) is a representation of the coding behavior and has a very large weakness of the code. If $_{PL}$ = 0, then (1) is an indication of an affine relationship in the code, but also a very large weakness. For Mod-2 acquisition systems, an affinity function is only complementary to a linear function. Both linear and affine approaches, indicated by $_{PL}$ 1/2 and $_{PL}$ i1/2, respectively, are equally sensitive to linear cryptanalysis, and will generally use the term linear to refer to both linear and affine relationships.

Given the non-linear properties of the S-box, it is possible to develop linear approaches between sets of input and output bits in the S-box. As a result, it is possible to combine the linear approaches of the S-boxes so that the intermediate bits (ie, the bits of data coming from the password) can be canceled and

obtained with a linear expression that has a large bias and contains only plain text.

First, it is necessary to make inferences about the fragility of S-box against linear cryptanalysis. Consider the S-box representation of Figure 7 with the input X = $_{[X1}X_2X_3X_4X_5X_6X_7X_8$] and a corresponding output Y = $_{[Y}$1Y2Y3Y4Y5Y6Y7Y8]. All linear approaches can be examined to determine their usefulness by calculating the probability bias for each. Therefore, we examine all expressions in the form of equation (1) where X and Y are S-box inputs and outputs, respectively [23].

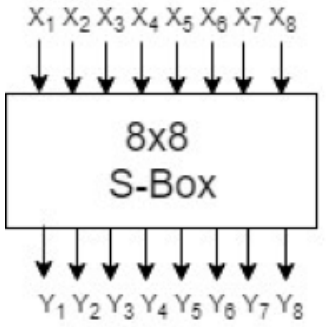


Figure 7. S-Box Structure

Considering the linear expression for the S-box used in the encryption algorithm;

$$X_1 \oplus X_2 \oplus X_3 \oplus X_4 \oplus X_5 \oplus X_6 \oplus X_7 \oplus X_8 \oplus \quad (3)$$

$$Y_1 \oplus Y_2 \oplus Y_3 \oplus Y_4 \oplus Y_5 \oplus Y_6 \oplus Y_7 \oplus Y_8 = 0$$

As an equivalent to the typed expression, the expression can be rewritten as follows:

$$X_1 \oplus X_2 \oplus X_3 \oplus X_4 \oplus X_5 \oplus X_6 \oplus X_7 \oplus X_8 \oplus \quad (4)$$

$$= Y_1 \oplus Y_2 \oplus Y_3 \oplus Y_4 \oplus Y_5 \oplus Y_6 \oplus Y_7 \oplus Y_8$$

In the encryption step, the number of S-Boxes that are actively used is taken into account. In the algorithm, 4 different S-Boxes will be used at different points in the rounds but at the same time, probabilities are calculated by creating a linear approximation table over a minimum of 4 active S-Boxes.

If the bias from 1/2 value of the linear approximation probability for the whole encryption algorithm is indicated by ε, the ε value is calculated as follows;

Table 3. Probability bias - highest bias values

Box Name	Bias Value
K Box	116/256 - 1/2 = (116-128)/256 = -12/256
L Box	136/256 - 1/2 = (136-128)/256 = 8/256
M Box	124/256 - 1/2 = (124-128)/256 = -4/256
N Box	22/256 - 1/2 = (122-128)/256 = -6/256

$$\varepsilon 1,2,..,n = 2^{n-1} \prod_{i=1}^{n} \varepsilon i \quad (5)$$

$$\varepsilon = \frac{2^{(4-1)}(-12)(8)(-4)(-6)}{256*256*256*256} \quad (6)$$

Matsui shows that the number of known plain texts required in the attack is proportional to $_{\varepsilon}2$ and that $_{NL}$ represents the required number of known plain texts, so it is possible to calculate $_{NL}$ approximately.

$$N_L = 1/\varepsilon^2 \quad (7)$$

In this case, the number of plain text-encrypted text pairs required for the attack is as follows:

$$N_L = \varepsilon^{-2} = [\frac{2^{32}}{2^{11}*9}]^2 = \frac{2^{42}}{9} \quad (8)$$

As a result from the calculated value for required plaintextcipher text pair the proposed encryption structure seems to durable to linear cryptoanalysis attacks.

Differential Cryptanalysis: Differential cryptanalysis uses the high likelihood of plaintext differences and certain occurrences of differences in the final round of the encryption algorithm. For example, when a system with output X =[$_{X1}X_2...{}_{X8}$] and Y = $_{[Y1}Y_2...{}_{Y8}$] is considered, the two inputs to the system are $_{X1}$ and $_{X}2$ with the corresponding outputs $_{Y1}$ and $_{Y2}$, respectively. The input difference is given by 4X = $_{X1} \oplus {}_{X2}$; wherein "⊕" represents the XOR processing of n-bit vectors.

4X = [4*X*114*X*2...*Xn*] operation; 4Xi = *X*1 ⊕ *X*2, and *Xi*1, $_{X}i2$ represent the first bit of variables $_{X1}$ and $_{X2}$. Similarly, Y= $_{[Y1}Y_2...{}_{Yn}$] indicates the difference 4 $_{Yi}$= $_{Y1} \oplus {}_{Y2}$.

Ideally in random encryption; for a given input difference of 4Y, the probability of the output difference resulting from the given value, such as 4X, is $_{1/2}n$, where n is the bit number of X. Differential cryptanalysis attempts to exploit a scenario in which a given input difference 4X and a given 4Y occur for a very high probability P_D value (ie greater than $_{1/2}n$). The resulting input/output pair (4X, 4Y) is called the differential.

Differential cryptanalysis is a selected plaintext attack, which means that the attacker can select the inputs and examine the outputs in an attempt to derive the key. For differential cryptanalysis, the attacker chooses input pairs $_{X1}$ and $_{X2}$ to meet a specific 4X, knowing that a specific 4Y value is likely to occur for this 4X value [23].

The differences were obtained by updating the related values in the table 4. Considering an ideal S-box, all values in the table should be expected to be 1, resulting in a 1/256 probability for each value. However, since the ideal S-box is not mathematically feasible, an S-Box should be used to give the best distribution.

Table 4. Difference distribution table of s-box with highest probabilities

Box Name	Highest Difference Value
K Box Table	10/256
L Box Table	12/256
M Box Table	10/256
N Box Table	12/256

Although it is difficult to say the exact number of plaintext and encrypted text pairs required to perform attacks, the following statement can calculate approximate values;

$$N_D = c/P_D \quad (9)$$

While the constant c in the expression is a small integer, the $_{PD}$ value indicates the characteristic differences of the differences obtained during the encryption process. This value is obtained as follows;

$$P_D = \prod_{i=1}^{\gamma} \beta_i \quad (10)$$

The γ symbol in the expression represents the active Sbox, while the β symbol indicates the characteristic probability of the difference. According to this;

$$P_D = c/((10*12*10*12)/256^4) =$$

$$= c/(14400/(2^8)^4) = c/(14400/2^{32}) = c/(1.4/2^{22}) \quad (11)$$

$$N_D = 2^{22}/1.4 = 2{,}995{,}931.4$$

Approximately 2,995,931 plain-text / encrypted-text pairs are required. This value is calculated with the lowest probability assuming 4 active S-Box values.

2) Performance Aspects: In IoT devices with limited resources, memory usage increases with the number of cycles in the algorithm. Therefore, the proposed algorithm should also be evaluated in terms of memory usage. According to the information shown in Table 5, it is seen that LWE algorithm does not perform better than other algorithms by using 296 bytes for encryption and 392 bytes for decoding. Another key parameter for the evaluation of the algorithm is the time taken to encrypt and decrypt a particular data. The proposed algorithm is designed to provide the IoT environment with minimal time and significant security. When the values shown in Table 5 are compared, encryption and decryption rates are performed better than other algorithms and ranked second in the table.

One method of observing the visual impact of the encryption algorithm is to encrypt an image with the proposed algorithm and observe the randomness it generates in the image. The his-

Table 5. Performans evaluation table

Cipher Name	Block Size (bit)	Key Size (bit)	Code Size E (byte)	Code Size D (byte)	RAM E (byte)	RAM D (byte)	Run Time E (byte)	Run Time D (byte)
Identity Cipher	64	80	238	356	32	192	558	842
LWE	64	64	1402	1595	296	392	1902	1850
Rectangle	64	80	452	587	40	128	3572	4090
LBlock	64	80	1165	1352	89	200	9886	10406
Road Runner	64	80	934	1087	136	232	14530	14744
TWINE	64	80	1104	1268	88	176	16356	17014
Piccolo	64	80	986	1178	160	304	16858	17100
PRESEN	T 64	80	3071	3522	168	312	22528	76242
LED	64	80	2402	2977	128	272	46872	67534

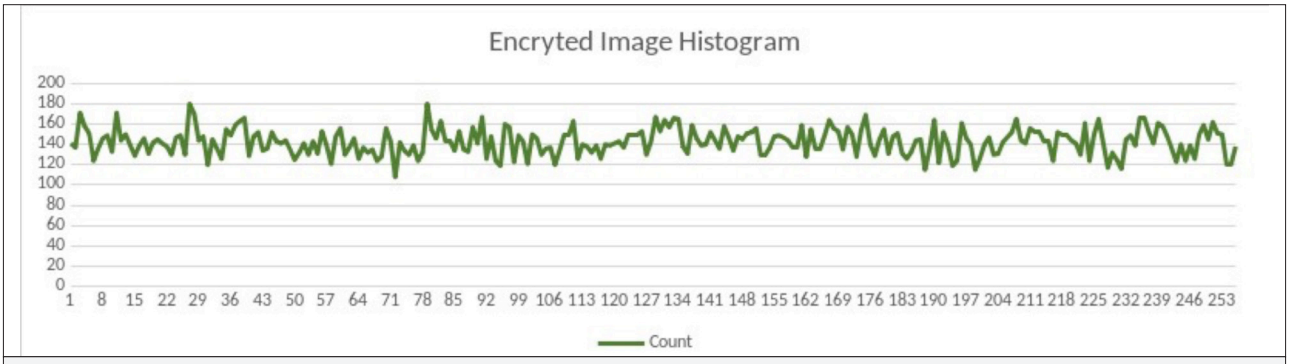


Figure 8. Histogram of Encrypted Image

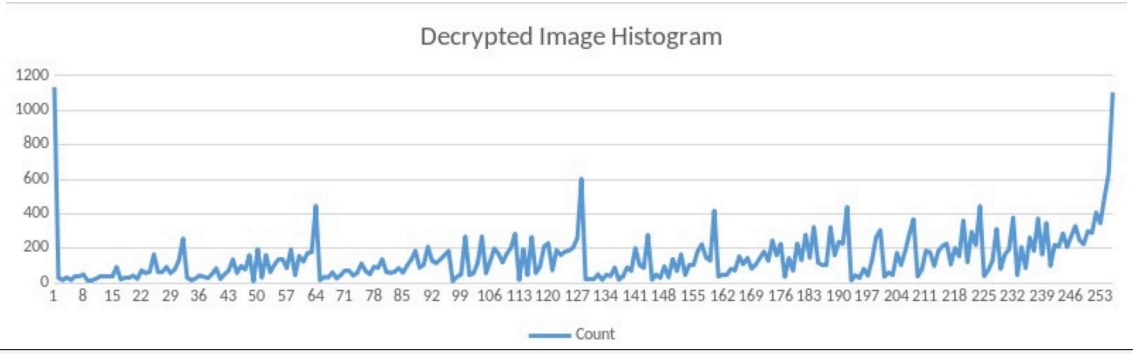


Figure 9. Histogram of Decrypted Image

togram of the image is calculated to evaluate the generated randomness. A homogeneous histogram after encoding shows the randomness success of the algorithm. Figure 9 shows the histogram value of an unencrypted image. The values shown in Figure 8 show the homogeneous distribution of an encrypted image. The proposed algorithm is shown to successfully distribute the histogram values homogeneously.

Conclusion

The proposed LWE block encryption algorithm has evaluated in terms of both security and performance aspects. The main purpose was to have a encryption algorithm that is light enough for restricted hardware environments and secure enough to endure primal cryptanalysis attacks. As a result of both security and performance evaluation, it is said to be that the LWE algorithm has a proper diffusion and confusion properties over the known plaintext-ciphertext attacks meaning that it has an adequate security endurance against known attacks. And the performance comparison with its equivalently purpose encryption algorithms, the result may said to be sufficient, even has better performance than other algorithms.

Peer-review: Externally peer-reviewed.

Conflict of Interest: The authors have no conflicts of interest to declare.

Financial Disclosure: This study is supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under grant no. EEEAG-117E579.

References

1. Mark Weiser. The Computer for the 21st Century. Scientific American, Feb, 1991. [CrossRef]
2. J. Gubbi, R. Buyya, S. Marusic, M. Palaniswam, "Internet of Things (IoT): A vision, architectural elements, and future directions", Future Generation Computer Systems vol. 29, no. 7, pp. 1645-60, Sep, 2013. [CrossRef]
3. H. Suo, J. Wan, C. Zou, J. Liu, "Security in the Internet of Things: A Review", 2012 International Conference on Computer Science and Electronics Engineering, 23-25 Mar, 2012. [CrossRef]
4. W. Wu, L. Zhang, "LBlock: A Lightweight Block Cipher", 9th International Conference, ACNS 2011, Nerja, Spain, 7-10 Jun, 2011.
5. M. Usman, I. Ahmed, M. I. Aslam, S. Khan, U. A. Shah, "SIT: A Lightweight Encryption Algorithm for Secure Internet of Things", IJACSA, Vol. 8, No. 1, pp. 402-11, 2017. [CrossRef]
6. A. Bogdanov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann, M.J.B. Robshaw, Y. Seurin, C. Vikkelsoe, "PRESENT: An Ultra-Lightweight Block Cipher", Available from: URL: http://www.lightweightcrypto.org/present/present_ches2007.pdf.
7. B. Schneier, "SIMON and SPECK: New NSA Encryption Algorithms". Schneier on Security. Available from: URL: https://www.schneier.com/blog/archives/2013/07/simon_and_speck.html.
8. R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, L. Wingers, "Simon and Speck: Block Ciphers for the Internet of Things", Available from: URL: https://eprint.iacr.org/2015/585.pdf.
9. D. J. WheelerRoger, M. Needham, "TEA, a Tiny Encryption Algorithm", In: Preneel B. (eds) Fast Software Encryption. FSE 1994. Lecture Notes in Computer Science, vol. 1008. Springer, Berlin, Heidelberg. [CrossRef]
10. Z. Gong, S. Nikova, Y. W. Law, "KLEIN: A New Family of Lightweight Block Ciphers", In: Juels A., Paar C. (eds) RFID. Security and Privacy. RFIDSec 2011. Lecture Notes in Computer Science, vol. 7055. Springer, Berlin, Heidelberg. [CrossRef]
11. Jesang Lee, Kitae Jeong, Hyun Kim, Jongsung Kim, Seongtaek Chee, "HIGHT: A New Block Cipher Suitable for Low Resource Device", In: Goubin L., Matsui M. (eds) Cryptographic Hardware and Embedded Systems - CHES 2006. CHES 2006. Lecture Notes in Computer Science, vol 4249. Springer, Berlin, Heidelberg.
12. J. Daemen, V. Rijmen, "The design of Rijndael: AES-the advanced encryption standard", Springer-Verlag Berlin Heidelberg, 2002. [CrossRef]
13. J. L. Massey, "SAFER K-64: A Byte-Oriented Block-Ciphering Algorithm", In: Anderson R. (eds) Fast Software Encryption. FSE 1993. Lecture Notes in Computer Science, vol. 809, Springer, Berlin, Heidelberg [CrossRef]
14. J. Daemen, "The Block Cipher Square", In: Biham E. (eds) Fast Software Encryption. FSE 1997. Lecture Notes in Computer Science, vol. 1267, Springer, Berlin, Heidelberg. [CrossRef]
15. N. Khatri, "Blowfish Algorithm", IJESMR, vol. 2, no. 10, pp. 45-51, Oct, 2015.
16. DES- Data Encryption Standard, International Research Journal of Engineering and Technology (IRJET).
17. S. Karthik, A. Muruganandam, "Data Encryption and Decryption by Using Triple DES and Performance Analysis of Crypto System", IJSER, Vol. 2, No. 1, pp. 24-31, Nov, 2014.
18. P. Sritha, R. Ashokkumar, S. Bhuvaneswari, M.Vidhya, "A new modified RC6 algorithm for cryptographic applications", IJARCCE Vol. 3, No. 12, Dec, 2014. [CrossRef]

19. R. Chandramouli, S. Bapatla, K.P. Subbalakshmi, "Battery Power-aware Encryption", ACM Transactions on Information and System Security, Volume. 9, No. 2, May, pp. 162-80, 2006. [CrossRef]
20. P. S. L. M. Barreto, V. Rijmen, "The Khazad legacy-level block cipher" Computer Science, Jan, 2000.
21. R. Forrié, "The Strict Avalanche Criterion: Spectral Properties of Boolean Functions and an Extended Definition, Advances in Cryptology" In: Goldwasser S. (eds) Advances in Cryptology - CRYPTO' 88. CRYPTO 1988. Lecture Notes in Computer Science, vol 403. Springer, New York, NY.
22. M. Matsui, "Linear Cryptanalysis Method for DES Cipher", In: Helleseth T. (eds) Advances in Cryptology - EUROCRYPT '93. EUROCRYPT 1993. Lecture Notes in Computer Science, vol 765. Springer, Berlin, Heidelberg.
23. H. M. Heys, "A tutorial on linear and differential cryptanalysis", Cryptologia Archive, Vol. 26, No. 3, Jul, pp. 189-221, 2002. [CrossRef]



Sezer Toprak has been working in Cyber Security field as Consultant in Deloitte Turkey for 2 years. He has been studying for Master of Science with thesis in Computer Engineering at Yildiz Technical University. He was graduated from Istanbul Ku¨ltu¨r University, Computer Engineering in 2017.



Akhan Akbulut is an associate professor in the Computer Engineering Department of Istanbul Ku¨ltu¨r University. His research interests focus on the design and performance optimization of software-intensive systems, Internet architectures, secure software development, and broadening participation in cloud computing research. Akbulut received a PhD in Computer Engineering from the Istanbul University, Turkey. Between 20017 and 2019, he worked as a postdoctoral researcher at the Computer Science Department of North Carolina State University



Muhammed Ali Aydin is currently working as an associate professor at Computer Engineering Department of Istanbul University-Cerrahpasia. His interest mainly focuses on Cyber Security, Cryptography, Network Security and Communication- Network Protocols. He received his PhD degree in Computer Engineering from Istanbul University and ha has completed Post Doctoral research at Telecom SudParis in Department of Computer Science.



Abdu¨l Halim Zaim is currently a faculty member in the Department of Computer Engineering at the Faculty of Engineering and Design at Istanbul Commerce University, a faculty member in the Department of Computer Engineering at the Faculty of Engineering at Istanbul University and the Director of the Center for Information Technology Application and Research at Istanbul Commerce University. Abdul Halim Zaim served as Vice Rector and Vice President of Academic Evaluation Commission at Istanbul Commerce University. He received his MS degree in Computer Engineering from Bogazici University in 1996 and his PhD in Electrical and Computer Engineering from North Carolina State University (NCSU) in 2001.