Statistical analysis of the key scheduling of the new lightweight block cipher

Nursulu Kapalova[1,2], Kunbolat Algazy[1,2], Armanbek Haumen[1], Kairat Sakan[1,2]
[1]Laboratory of Information Security, Institute of Information and Computational Technologies, Almaty, Republic of Kazakhstan
[2]Faculty of Information Technology, Al-Farabi Kazakh National University, Almaty, Republic of Kazakhstan

Article Info

Article history:

Received Apr 26, 2023
Revised Jul 18, 2023
Accepted Jul 3, 2023

Keywords:

Cryptography
Encryption algorithm
Lightweight algorithm
Round key generation algorithm
Round keys

ABSTRACT

This research paper is aimed at studying the generation of round keys (GRK) of the lightweight block cipher (LBC), which provides an optimal balance between security, performance, and minimal costs in internet of things (IoT). For comparative analysis, the GRK of the well-known PRESENT algorithm was studied. A number of studies have been carried out to assess the cryptographic strength of encryption algorithms, however, less attention has been paid to the assessment of the reliability of GRK algorithms, which can lead to a possible weakening of a cryptosystem. A trusted GRK should issue random and independent round keys regardless of the secret key. The experiments were carried out with secret keys of low and high density, as well as random numbers. The obtained results show that the GRK of the LBC algorithm generates random round keys that successfully pass tests of the National Institute of Standards and Technology (NIST) for randomness.

Corresponding Author:

Kunbolat Algazy
Laboratory of Information Security, Institute of Information and Computational Technologies
28 Shevchenko, Almaty, 050010, Republic of Kazakhstan
Email: kunbolat@mail.ru

1. INTRODUCTION

Any cipher is a certain transformation of the plaintext, which depends on the key. Modern block ciphers include a key schedule algorithm to obtain an array of round keys from a secret key and an encryption algorithm. The security of a cipher depends on both algorithms and is judged by the quality and length of the encryption key. According to National Institute of Standards and Technology (NIST) requirements [1], [2], the minimum length of the encryption key should be 112 bits. In block algorithms, the degree of security of encrypted data directly depends on the transformations used and the algorithm for generating round keys (GRK) [3]. If the GRK provides a complex dependence of the round keys on the secret key, then many cryptanalysis methods will be inefficient.

Round keys are directly added to the original encryption data to hide the dependency between input and output in a particular round. Therefore, the statistical properties of the key schedule scheme are important for obtaining random ciphertexts in each round [4], [5]. When developing an encryption algorithm, it is impossible to test all combinations of master keys. However, there is a strong possibility that deliberate exploration or accidental discovery may reveal some of the secret keys, leading to an unexpected weakening of a cipher [6]. Such secret keys are considered weak keys. Thus, the security of a GRK algorithm is just as important as the encryption to eliminate weak keys. Therefore, when developing an encryption algorithm, it is required that round keys are created using a special round key generation algorithm and depend on the values of most bits of the original encryption key [7], [8].

Compared to the requirements for the strength of an encryption algorithm, less attention has been paid in the literature to the requirements for the reliability of a GRK algorithm. Knudsen and Mathiassen [9] indicate that the GRK is secure if all round keys are equally protected. According to Poojari and Nagesh [7], several attacks can use a linear relationship between round key bits. According to the authors, a GRK should have the property of an avalanche effect and all bits of the input keys should have the same effect on the output round keys. Due to this, one can achieve the maximum avalanche effect in round keys. Kelsey *et al.* [10] show that by excluding linearity in the GRK and providing an avalanche and a strict avalanche effect in round keys, it is possible to prevent attacks on related keys and differential attacks.

May *et al.* [11] proposes three desirable properties for a GRK, namely a one-way anti-collision function, independence between round and secret keys, and efficient implementation. In [12], an estimate of the independence between the round keys generated by the GRK is presented. The authors have shown that the independence between round keys ensures the resistance of the GRK to a key-dependent attack.

The reliability of the GRK algorithm depends on the functions used to generate round keys. Numerous GRK algorithms are proposed in scientific papers, which use various methods to generate round keys. For example, the block cipher international data encryption algorithm (IDEA) uses a linear function (rotate shift) to generate round keys [13]. Similarly, the GRK algorithm of the PRESENT block cipher uses a linear permutation as its main component to generate round keys [14]. However, in the PRESENT cipher, a non-linear function (s-box) is only used for a limited number of bits (only the outermost 4 bits). The GRK from the advanced encryption standard (AES) algorithm uses linear and non-linear functions to generate round keys [15].

Sulaiman *et al.* [16] argue that a quality key schedule should be developed using those transformations that are used in the main part of the cipher. This approach is used for the key schedule of the lightweight Serpent encryption algorithm [17]. To evaluate the reliability of the lightweight block cipher (LBC) GRK algorithm, a frequency test was performed, as well as tests of bit independence and bit correlation for high/low-density keys. In this article, for comparative studies, the GRK of the PRESENT cipher is considered.

The purpose of the research work is to evaluate the statistical security of the GRK algorithm of the LBC cipher. It includes the study of the cryptographic properties of the output sequence of the round key generation algorithm using statistical tests. If the GRK algorithm produces poor results in statistical tests, then it will certainly not be resistant even to basic attacks [18], [19].

2. METHOD

2.1. Block encryption algorithm LBC

The LBC algorithm is designed to encrypt 64-bit block-type data with an 80-bit key. LBC encrypts for 20 rounds. Each round includes 4 types of transformation [20]: i) *S* transformation, ii) *RL* transformation, iii) *L* transformation, and iv) *K* transformation.

During encryption, the original plaintext block is divided into 4 subblocks of 16 bits each. Encryption starts by adding the first 64 bits of the master key to the plain text as shown in Figure 1. Next, round transformations are performed.

If we designate the round encryption process as *E*, we will get:

$$E(A) = [K(L(S(A_0)||RL(S(A_0)) \oplus S(A_1)||S(A_2)||S(A_3)))]_{rc} = B,$$

where $A = \{A_0, A_1, A_2, A_3\}$ is the 64-bit plaintext, $\{A_i\}$ are 16-bit subblocks, *B* is the 64-bit ciphertext, *rc* is the number of rounds. The *RL* function is executed only for the subblock $\{A_0\}$. That is $\{A_1\} = RL(A_0) \otimes A_1$. Nyssanbayeva *et al.* [20] provides a detailed description of this encryption algorithm.

2.2. Key scheduling in encryption algorithms

Let *Y* be a key of a symmetric block encryption algorithm. Then a key schedule is the system of functions $\{\tau_1, \tau_2, \dots, \tau_r\}$, which allows extending the key *Y* to *r* keys $y_1, \dots, y_r$, where *r* is the number of rounds of the iterative (round) block cipher. A key schedule is a process of generating a key sequence based on a master key (secret key). The same numerical sequences, so-called round keys, will be used for round encryption procedures [21].

2.2.1. Key schedule in the LBC algorithm

The length of the secret key (master key) of the LBC algorithm is 80 bits. From this secret key, the round keys of the algorithm are generated. The key schedule in the considered algorithm includes several transformations, which are presented in Figure 2.

A master key of 80 bits is divided into 5 subblocks of 16 bits each. In the first step, the bits of the first key subblock are replaced with other bits using the S-box substitution table. The fourth subblock is added with the constant *Nr* modulo 2. Here *Nr* is the number of a round, which takes values from 1 to 20. In the second step, the bits of each subblock rotate left by several positions. This procedure is performed by the function *R_i*, where *i* is the number of positions by which the bits of the subblock rotate. As shown in Figure 2, the bits of the first subblock rotate left by 6 positions, the second subblock-by 7, the third subblock-by 8, and the remaining subblocks-by 9 and 10 positions respectively. In the third step, the subblocks are summed by XOR operations with the neighboring right subblock, i.e., the first with the second, the second with the third, and the third with the fourth subblock. In the last step of the transformation, the subblocks rotate left by one position, that is, the entire subblocks are swapped.

After the above transformations, the initial 4 sub-blocks (64 bits) of the sequence of 5 sub-blocks (80 bits) starting from the left are taken and then used as the encryption (decryption) round keys. The 80 bits obtained from the previous transformations will be used as the basis for the next round keys. This process continues until all round encryption keys are received.

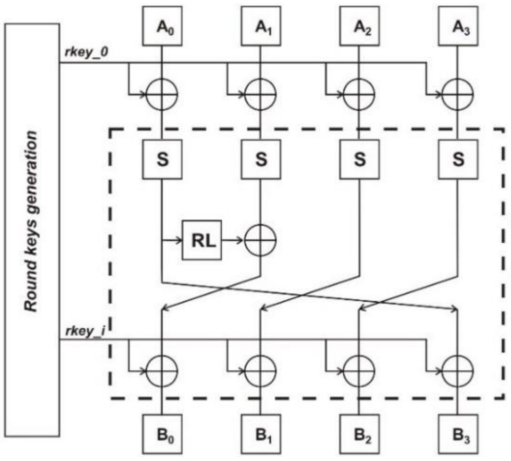

Figure 1. General scheme of the LBC algorithm encryption process

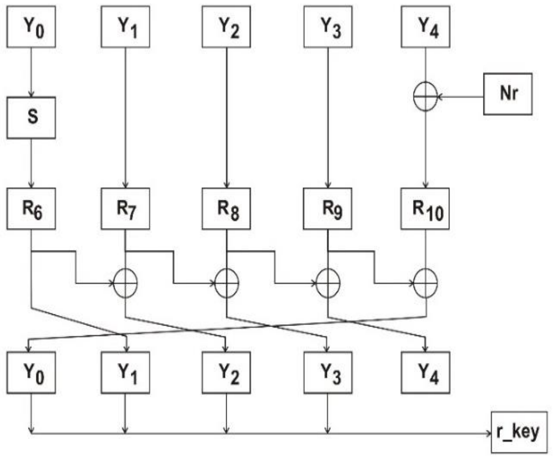

Figure 2. Scheme for generating round keys of the LBC algorithm

2.2.2. Key schedule in the PRESENT algorithm

The PRESENT algorithm can accept keys of either 80 or 128 bits [22]. However, this paper will consider key scheduling with 80-bit master keys. The secret 80-bit key is stored in the key register Y and represented as a sequence of bits $\{y_{79}, y_{78}, \dots, y_0\}$. The first-round key is formed from 64 leftmost bits of the main secret key $Y_i = \{y_{63}, y_{62}, \dots, y_0\}$ consists of the 64 leftmost bits of the current contents of the register Y. So, after the first round we have:

$$Y_i = \{y_{63}, y_{62}, \dots, y_0\} = \{y_{79}, y_{78}, \dots, y_{16}\}$$

After extracting the round key Y_i, the key register $Y = \{y_{79}, y_{78}, \dots, y_0\}$ is updated as:

$$[y_{79}, y_{78}, \dots, y_1, y_0] = [y_{18}, y_{17}, \dots, y_{20}, y_{19}],$$
$$[y_{79}, y_{78}, y_{77}, y_{76}] = S[y_{79}, y_{78}, y_{77}, y_{76}],$$
$$[y_{19}, y_{18}, y_{17}, y_{16}, y_{15}] = [y_{19}, y_{18}, y_{17}, y_{16}, y_{15}] \oplus round\_counter.$$

Thus, the key register rotates left by 61 positions, then the leftmost four bits are passed through the S-box of the main encryption algorithm. The *round_counter* value is added to the $y_{19}y_{18}y_{17}y_{16}y_{15}$ bits from the key register *Y*. These procedures are repeated 31 times until the last round key is obtained.

2.3. Evaluation

This section discusses key schedule estimates in the LBC algorithm. The following data types were selected for testing: high-density key (HDK) sequence, low-density key (LDK) sequence, and random density key (RDK) sequence. The HDK sequence consists mainly of ‘1’ bits and contains only one or two ‘0’ bits. On the contrary, LDK consists mostly of ‘0’ bits and contains only one or two ‘1’ bits. RDK is a random sequence of bits. Such a scheme of initial data was taken from [19].

In this paper, we give an estimate of the quality of the GRK of the LBC algorithm. To compare the results, work was also carried out to evaluate the key schedule of the PRESENT-80 algorithm. All initial data

(HDK, LDK, and RDK) are 80 bits long and will serve as a secret key for the GRK of the LBC and PRESENT algorithms. The following tests were used to assess the quality and reliability of the key schedule of the algorithms: frequency test, bit difference between round keys, Hamming weight test, and avalanche effect in round keys.

2.3.1. Frequency test

The focus of the test is on the ratio of zeros and ones for the entire sequence. For numeric key data generated by round keys, the number of 1s and 0s in the sequence should be approximately equal. Thus, the frequency test is the basic NIST test for proving the randomness of numerical sequences in round keys [23], [24]. If any GRK fails to produce random round keys, then there is no need to conduct any further tests since they impose more stringent requirements, and the algorithm will certainly not comply with them [19]. An estimate in the frequency test can be obtained using formulas (1)-(3). To conduct a frequency test, we perform the following operations [23]:

a. Convert the zeros and ones of the input sequence to -1 and +1, respectively, and sum to obtain

$$S_n = X_1 + X_2 + \cdots + X_n, \text{ where } X_i = 2\varepsilon_i - 1, \tag{1}$$

where ε_i are the bits of the input sequence.

b. Calculate test statistics:

$$S_{obs} = \frac{|S_n|}{\sqrt{n}}. \tag{2}$$

c. Calculate p-value:

$$p - value = erfc\left(\frac{S_{obs}}{\sqrt{2}}\right); \tag{3}$$

where $erfc$ is the complementary error function.

To estimate the key schedule of both algorithms, we used 10,000 secret keys. All these keys were combined into one file. In each round, the keys are also written to one file. Since the LBC algorithm consists of 20 rounds of encryption, 20 round keys were taken for testing, that is, 20 separate files with a round key in each. To compare the test results, 20 round keys were also taken from the PRESENT algorithm. Each file was individually tested with the frequency test.

The frequency test evaluates the calculated *p*-value from the input bit sequence. The *p*-value indicates the probability of obtaining the observed results given that the null hypothesis is true, or the probability of error if the null hypothesis is rejected. Based on the calculated *p*-value, it is decided that if the *p*-value is ≥0.01, then the sequence is considered pseudo-random, otherwise it is non-random [23].

2.3.2. Bit difference between round keys

The purpose of this test is to identify the complex relationship between round keys [19] by assessing confusion in round keys. Two consecutive round keys are summed by the XOR operation and the total number of 1 s in the resulting vector is calculated. A reliable key schedule provides a bit difference of 50%, which greatly complicates the statistical relationship between the secret key and the round keys.

2.3.3. Hamming weight test

Hernandez-Castro *et al.* [25] performed a probabilistic metaheuristic search. In doing so, they identified a set of round keys that had very low Hamming weight. In differential cryptanalysis, such keys will be an ideal basis for attacking the encryption algorithm. A perfectly balanced binary sequence of *n* bits should have a Hamming weight of *n*/2 [25].

In the LBC algorithm, only the leftmost 64 bits of the key sequence are used for the encryption process in each round, and the remaining bits are not considered during the encryption process. Thus, the Hamming weight of round keys is calculated for these 64 bits. Since, in our case, the value of *n* is 64, then the ideal value of the Hamming weight for each sequence is *n*/2=32. The LBC algorithm uses 20 round keys. Therefore, the value of the Hamming weight should be 20 rounds×32 bits=640 bits. This is 50% of the total number of bits. In this test, the same key is used as the secret key for both the LBC algorithm and the Present algorithm, and the Hamming weight for each round key is calculated by (4):

$$Hamming\ weight = \frac{Number\ of\ 1\ bits}{Total\ number\ of\ bits} \tag{4}$$

2.3.4. Avalanche effect in round keys

The avalanche effect (AE) in encryption algorithms estimates the degree to which the ciphertext changes when one-bit changes in the corresponding plaintext or secret key. This test determines the nonlinear characteristics of the transformations used in the proposed algorithm. The best avalanche effect is 50% or more, ensuring that changing any bit of the secret key affects the bits of the ciphertext [26].

The avalanche effect is calculated using (5):

$$AE = \frac{1}{x}\sum_{i=1}^{x}|c_i - p_i| \tag{5}$$

where x is the length of the plaintext (ciphertext), c_i is the ith bit of the ciphertext, and p_i is the ith bit of the plaintext. The resulting value is converted into a percentage for comparison using (6).

$$percentage\ of\ bits\ changed = AE \times 100. \tag{6}$$

For this criterion, the sequences LDK, HDK, and RDK described in subsection 2.3 were used. At the beginning of testing, based on the selected key, the round keys K_i'. are generated. Then we change one bit of the original key and generate the round keys K_i'.

$$K_i' = \{k_1', k_2', \dots, k_{rc}'\}; \tag{7}$$

Where i is the position of the changed bit of the original key and rc is the sequential number of the round.

Using the formula (5), we find the values of AE for each round. In each iteration, we change the next bit of the original key and find new values of AE. This method was used to study the round keys of the LBC and PRESENT-80 algorithms.

3. RESULTS AND DISCUSSION

3.1. Frequency test

The results of the frequency test are considered to be the decisive factor for the key sequence generated by the GRK of the encryption algorithm. If the key sequence passes the frequency test, then the round keys are considered random. Otherwise, there is no need to perform other statistical tests.

A set of 10,000 random secret keys was used to test the GRK of both studied algorithms. The key sequences generated by the GRK of the LBC and PRESENT algorithms pass the frequency test. Since the GRK input data were random, the generated round key sequences were also random. This indicates that the GRK of the LBC and PRESENT algorithms generate key sequences that are close to random.

Table 1 shows the pass rate of 20 round keys obtained from random secret sequences using the GRK of the algorithms under study. The average *p-value* of the LBC algorithm is ~0.58, while that of the PRESENT algorithm is ~0.46. These results show that if the secret key is random, then a good frequency can be achieved using round keys.

Table 1. Average *p*-value of round keys

Round	LBC	PRESENT	Round	LBC	PRESENT
1	0.82	0.88	11	0.15	0.55
2	0.19	0.84	12	0.99	0.32
3	0.52	0.66	13	0.90	0.30
4	0.05	0.50	14	0.36	0.67
5	0.83	0.44	15	0.32	0.43
6	0.45	0.49	16	0.95	0.31
7	0.59	0.34	17	0.83	0.15
8	0.50	0.34	18	0.60	0.12
9	0.91	0.25	19	0.18	0.43
10	0.39	0.34	20	0.99	0.88

3.2. Bit difference between round keys

As in other algorithms, in LBC round keys are summed with round data modulo 2 (XOR operation). Therefore, if the round keys are pseudo-random, then the encrypted data, when summed with such keys, will be less predictable and more random. The test evaluates the average value of the bit difference for random secret keys. For testing, three types of sequences for the secret key were used: LDK, HDK, and RDK. 50 secret keys of each type were chosen.

Tables 2 and 3 show the percentage of bit differences between two consecutive round keys generated by the LBC GRK and the PRESENT GRK, respectively. As the results of the study show, for the LBC algorithm, the average value of the differences of the round keys of three types is approximately the same: LDK-49.32%, HDK-49.97%, and RDK-50.41%. At the same time, the PRESENT algorithm has these results: LDK-33.05%, HDK-34.44%, and RDK-49.87%. With keys of the LDK type, the percentage of bit difference ranges from 39% to 51% for the LBC algorithm and from 11% to 52% for the PRESENT algorithm. A good result was shown by keys of the RDK type: from 49% to 51% for both algorithms. Ideally, the bit difference between round keys should be at least 50%. Weak bit changes in the initial rounds can help attackers predict the keys of previous rounds and gain access to the secret key. Figure 3 shows the average percentage bit differences between the generated round keys of the LBC and PRESENT algorithms.

Table 2. Results of calculating the differences between the round keys of the LBC algorithm

y(i) XOR y(i+1)	LBC						
	HDK		LDK		Random Keys		Average
	Number of Bit Diff.	% of Bit Diff.	Number of Bit Diff.	% of Bit Diff.	Number of Bit Diff.	% of Bit Diff.	
y1+y2	35.96	44.95%	31.2	39.00%	41.08	51.35%	45.10%
y2+y3	40.86	51.08%	36.4	45.50%	40.02	50.03%	48.87%
y3+y4	38.66	48.33%	40.06	50.08%	40.82	51.03%	49.81%
y4+y5	40.54	50.68%	39.98	49.98%	41.32	51.65%	50.77%
y5+y6	39.64	49.55%	39.6	49.50%	39.94	49.93%	49.66%
k6+y7	40.4	50.50%	39.9	49.88%	40.12	50.15%	50.18%
y7+y8	40.6	50.75%	39.8	49.75%	40.64	50.80%	50.43%
y8+y9	40.22	50.28%	40.5	50.63%	40.02	50.03%	50.31%
…	…	…	…	…	…	…	…
y18+y19	41.32	51.65%	40.28	50.35%	39.68	49.60%	50.53%
y19+y20	40.06	50.08%	40.94	51.18%	39.82	49.78%	50.34%
Average	**39.98**	**49.97%**	**39.45**	**49.32%**	**40.33**	**50.41%**	**49.90%**

Table 3. Results of calculating the differences between the round keys of the PRESENT algorithm

k(i) XOR k(i+1)	PRESENT						
	HDK		LDK		Random Keys		Average
	Number of Bit Diff.	% of Bit Diff.	Number of Bit Diff.	% of Bit Diff.	Number of Bit Diff.	% of Bit Diff.	
y1+y2	10.1	12.63%	9.3	11.63%	40.98	51.23%	25.16%
y2+y3	11.04	13.80%	10.22	12.78%	40.28	50.35%	25.64%
y3+y4	14.56	18.20%	14.58	18.23%	38.28	47.85%	28.09%
y4+y5	14.7	18.38%	13.82	17.28%	40.04	50.05%	28.57%
y5+y6	18.1	22.63%	16.4	20.50%	39.44	49.30%	30.81%
y6+y7	18.16	22.70%	17.34	21.68%	40	50.00%	31.46%
y7+y8	23.36	29.20%	22.54	28.18%	40.12	50.15%	35.84%
y8+y9	23.28	29.10%	22.42	28.03%	39	48.75%	35.29%
…	…	…	…	…	…	…	…
y18+y19	41.28	51.60%	39.34	49.18%	40.18	50.23%	50.33%
y19+y20	44.8	56.00%	41.88	52.35%	39.7	49.63%	52.66%
Average	**27.55**	**34.44%**	**26.44**	**33.05%**	**39.90**	**49.87%**	**39.12%**

3.3. Hamming weight test

Round keys with low Hamming weights can be helpful in differential cryptanalysis attacks, as discussed earlier. For testing, again, 50 secret keys of each of the three types were selected: LDK, HDK, and RDK. Based on these secret keys, round keys were generated. For each key, the Hamming weight was calculated using (4). The results obtained are shown in Table 4. With the LDK and HDK keys, the LBC algorithm deviates from the optimal value of the Hamming weight by only 1.49% and 1.31%, while the PRESENT algorithm deviates much more -19.74% and 19.75%, respectively. With RDK keys, both algorithms take approximately the same values as shown in Table 4. It can be seen from the results that the LBC algorithm generates round keys with the Hamming weight from 48% to 50% for any type of secret key as shown in Figure 4.

3.4. Avalanche effect in round keys

To test the avalanche effect in round keys, one key of each of the HDK, LDK, and RDK types was selected. First, round keys were generated from the master key using the GRK. In the second step, we changed the first bit of the master key and again generated round keys. The resulting two groups of round keys were compared using (5), but here we were looking for the difference between the bits of the two groups of round keys. In each next step, the next bit of the master key was changed. In this way, indicators of the avalanche effect were calculated for each changed bit position of the master key. The test was carried out for each of the two algorithms separately. According to the test results, the avalanche indicator of the round keys of the LBC algorithm averaged ≈37%, while the same indicator of the PRESENT algorithm averaged ≈2%. The average values of the avalanche effect are shown in Table 5.

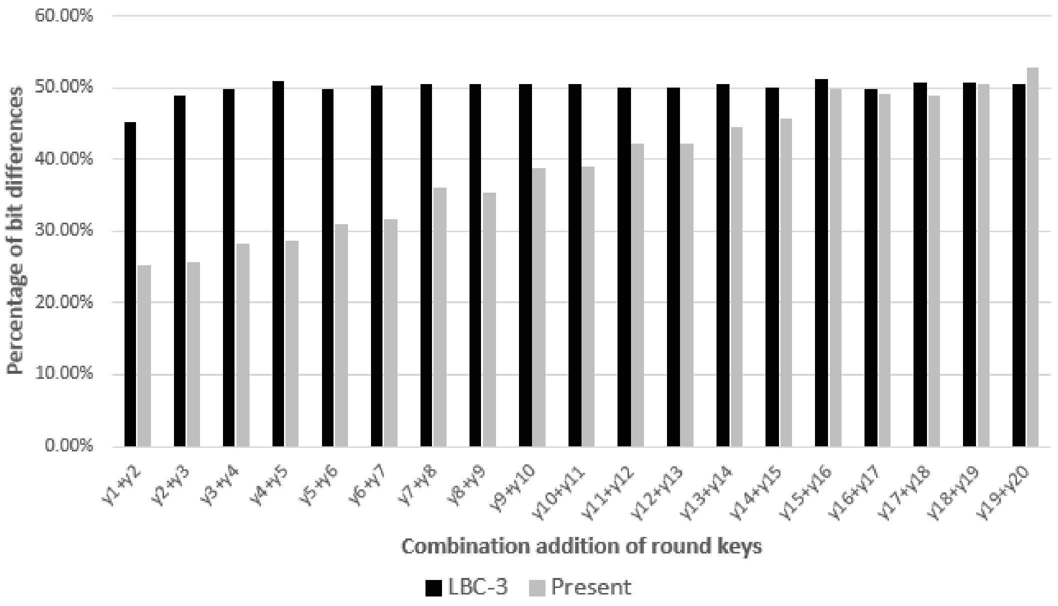

Figure 3. Average values of the percentage of bit differences between the generated round keys of the LBC and PRESENT algorithms

Table 4. Average value of the Hamming weight of round keys

Key type	LDK		HDK		RDK	
Algorithm	LBC	PRESENT	LBC	PRESENT	LBC	PRESENT
Average value of Hamming weight	48.51%	30.26%	48.69%	69.75%	50.08%	49.95%

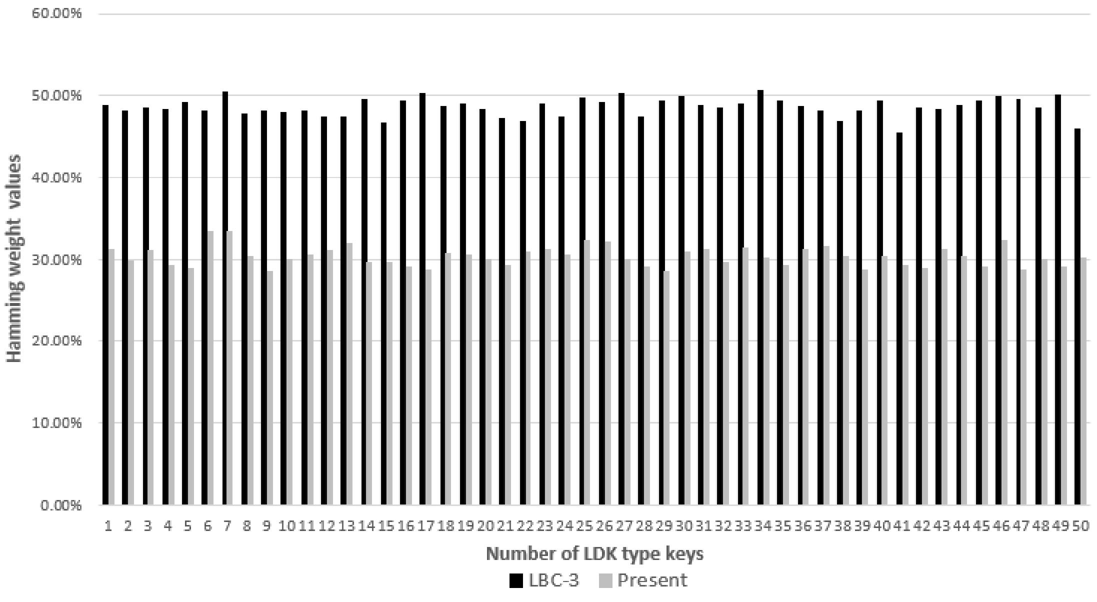

Figure 4. Average values of the Hamming weight of the LDK type round keys

Table 5. Average values of the avalanche effect of round keys

Algorithm	HDK	LDK	RDK
LBC	37.39%	37.45%	37.50%
PRESENT	2.24%	2.25%	2.21%

Table 5 shows that the avalanche effect in the round keys of the PRESENT algorithm is very low. This is because the non-linear transformation in the key schedule of this algorithm has very little effect on the bits of the key sequence. In the key schedule of the PRESENT algorithm, only the 4 extreme bits of the key sequence pass through the S-box (substitution table) in each round. Besides, beats 15 to 19 are added to the round number. This process also reduces the influence of the key sequence on the avalanche effect. In the LBC algorithm, the S-box operates on 16 bits in each round, and after rotate shifts, each block passes through the S-box. Therefore, the avalanche effect of this algorithm is higher than that of the PRESENT algorithm as shown in Figures 5, 6, and 7.

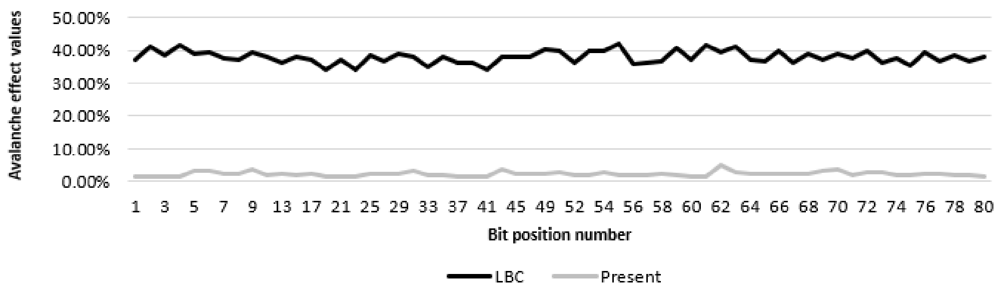

Figure 5. Average values of the avalanche effect of HDK-type round keys

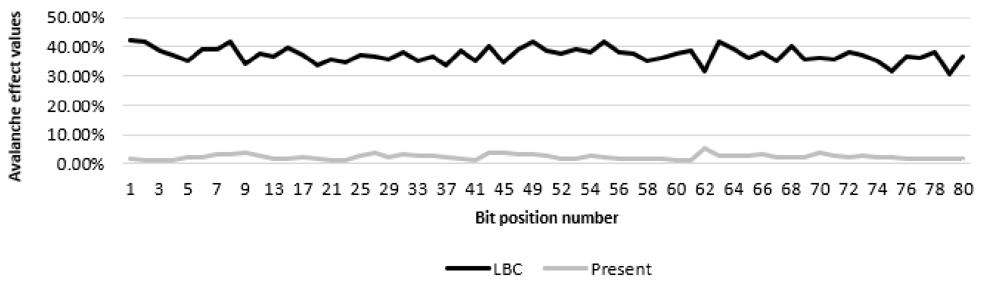

Figure 6. Average values of the avalanche effect of LDK-type round keys

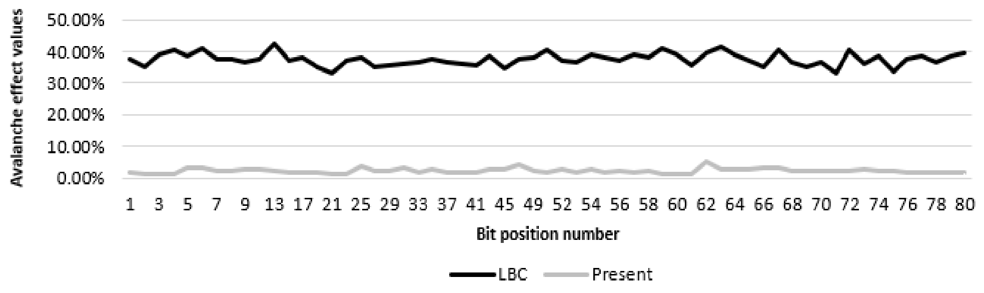

Figure 7. Average values of the avalanche effect of RDK-type round keys

In this test, we calculated avalanche effect values only for round keys. In [19], the same test was carried out together with the original text. Obviously, with such tests, the avalanche rate will be higher than that of tests for round keys.

4. CONCLUSION

This study aims to investigate the security impact of existing key scheduling procedures in the lightweight LBC and PRESENT symmetric block cipher algorithms. The results show that the GRK of the LBC algorithm demonstrates good statistical performance. The PRESENT algorithm GRK provides random round keys only when the secret key is random, but the LBC key schedule provides round key randomness for low-density keys, high-density keys, and random secret keys. The key sequence generated by the GRK of the LBC algorithm has completely passed the frequency test. The average value of the percent differences in bits between the generated round keys of the LBC algorithm is approximately 50%, which proves the good quality of the key schedule. The Hamming weight of the round keys of the algorithm is also within the normal range. The indicators of the avalanche effect of the round keys of the LBC algorithm had an average value of ≈37%. This means that the key schedule of the LBC algorithm is provided with a sufficient level of security for the encryption algorithm itself.

ACKNOWLEDGMENT

The research work was carried out within the framework of project AP14870719 "Development and study of post-quantum cryptography algorithms based on hash functions" at the Institute of Information and Computational Technologies of the RK MSHE CS.

REFERENCES

[1] E. Barker and A. Roginsky, "Transitions: Recommendation for transitioning the use of cryptographic algorithms and key lengths," *NIST Special Publication*, vol. 800, 2011.

[2] E. Barker and A. Roginsky, "Transitioning the use of cryptographic algorithms and key lengths," National Institute of Standards and Technology, Gaithersburg, MD, Mar. 2019. doi: 10.6028/NIST.SP.800-131Ar2.

[3] S. Afzal, M. Yousaf, H. Afzal, N. Alharbe, and M. R. Mufti, "Cryptographic strength evaluation of key schedule algorithms," *Security and Communication Networks*, pp. 1–9, May 2020, doi: 10.1155/2020/3189601.

[4] N. Kapalova, K. Sakan, K. Algazy, and D. Dyusenbayev, "Development and study of an encryption algorithm," *Computation*, vol. 10, no. 11, Nov. 2022, doi: 10.3390/computation10110198.

[5] R. G. Biyashev, S. E. Nyssanbayeva, and Y. Y. Begimbayeva, "Development of the model of protected cross-border information interaction," *Open Engineering*, vol. 6, no. 1, Aug. 2016, doi: 10.1515/eng-2016-0025.

[6] G. Marinakis, "Selection of sampling keys for cryptographic tests," *Cryptology ePrint Archive*, 2021.

[7] A. Poojari and H. R. Nagesh, "FPGA implementation of random number generator using LFSR and scrambling algorithm for lightweight cryptography," *International Journal of Applied Science and Engineering*, vol. 18, no. 6, pp. 1–9, 2021, doi: 10.6703/IJASE.202112_18(6).001.

[8] K. Algazy, K. Sakan, and N. Kapalova, "Evaluation of the strength and performance of a new hashing algorithm based on a block cipher," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 13, no. 3, pp. 3124–3130, Jun. 2023, doi: 10.11591/ijece.v13i3.pp3124-3130.

[9] L. R. Knudsen and J. E. Mathiassen, "On the role of key schedules in attacks on iterated ciphers," in *Computer Security ESORICS 2004*, Springer Berlin Heidelberg, 2004, pp. 322–334.

[10] J. Kelsey, B. Schneier, and D. Wagner, "Key-schedule cryptanalysis of IDEA, G-DES, GOST, SAFER, and triple-DES," in *Advances in Cryptology CRYPTO '96*, Springer Berlin Heidelberg, 1996, pp. 237–251.

[11] L. May, M. Henricksen, W. Millan, G. Carter, and E. Dawson, "Strengthening the key schedule of the AES," in *Information Security and Privacy*, Springer Berlin Heidelberg, 2002, pp. 226–240.

[12] A. C. Cakil and F. Ozkaynak, "A hybrid key schedule algorithm based on blum blum shub generator and chaotic systems," in *2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT)*, Sep. 2021, pp. 259–262, doi: 10.1109/CSIT52700.2021.9648671.

[13] J. Daemen, R. Govaerts, and J. Vandewalle, "Weak keys for IDEA," in *Advances in Cryptology-CRYPTO' 93*, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 224–231.

[14] A. Bogdanov *et al.*, "PRESENT: an ultra-lightweight block cipher," in *Cryptographic Hardware and Embedded Systems-CHES 2007*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 450–466.

[15] J. Daemen and V. Rijmen, *The design of Rijndael*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002.

[16] S. Sulaiman, Z. Muda, and J. Juremi, "The new approach of Rijndael key schedule," in *Proceedings Title: 2012 International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec)*, Jun. 2012, pp. 23–27, doi: 10.1109/CyberSec.2012.6246100.

[17] E. Biham, R. Anderson, and L. Knudsen, "Serpent: A new block cipher proposal," in *Fast Software Encryption*, Springer Berlin Heidelberg, 1998, pp. 222–238.

[18] R. E. J. Paje, A. M. Sison, and R. P. Medina, "Multidimensional key RC6 algorithm," in *Proceedings of the 3rd International Conference on Cryptography, Security and Privacy*, Jan. 2019, pp. 33–38, doi: 10.1145/3309074.3309095.

[19] M. Imdad, S. N. Ramli, and H. Mahdin, "An enhanced key schedule algorithm of PRESENT-128 block cipher for random and non-random secret keys," *Symmetry*, vol. 14, no. 3, Mar. 2022, doi: 10.3390/sym14030604.

[20] S. Nyssanbayeva, N. Kapalova, A. Haumen, and O. Suleimenov, "The LBC-3 lightweight encryption algorithm," *Open Engineering*, vol. 12, no. 1, pp. 570–577, Oct. 2022, doi: 10.1515/eng-2022-0372.

[21] B.-T. Liu, L. Li, R.-X. Wu, M.-M. Xie, and Q. P. Li, "Loong: A family of involutional lightweight block cipher based on SPN structure," *IEEE Access*, vol. 7, pp. 136023–136035, 2019, doi: 10.1109/ACCESS.2019.2940330.

[22] S. Banik, A. Bogdanov, T. Isobe, and M. Jepsen, "Analysis of software countermeasures for whitebox encryption," *IACR Transactions on Symmetric Cryptology*, pp. 307–328, Mar. 2017, doi: 10.46586/tosc.v2017.i1.307-328.

[23] A. Rukhin *et al.*, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," National Institute of Standards and Technology, Gaithersburg, MD, 2000. doi: 10.6028/NIST.SP.800-22.

[24] R. G. Biyashev, N. A. Kapalova, D. S. Dyusenbayev, K. T. Algazy, W. Wojcik, and A. Smolarz, "Development and analysis of symmetric encryption algorithm Qamal based on a substitution-permutation network," *International Journal of Electronics and Telecommunications*, Jul. 2023, doi: 10.24425/ijet.2021.135954.

[25] J. C. Hernandez-Castro, P. Peris-Lopez, and J.-P. Aumasson, "On the key schedule strength of PRESENT," in *Data Privacy Management and Autonomous Spontaneus Security*, Springer Berlin Heidelberg, 2012, pp. 253–263.

[26] O. C. Abikoye, A. D. Haruna, A. Abubakar, N. O. Akande, and E. O. Asani, "Modified advanced encryption standard algorithm for information security," *Symmetry*, vol. 11, no. 12, Dec. 2019, doi: 10.3390/sym11121484.

BIOGRAPHIES OF AUTHORS

Nursulu Kapalova received her master's degree in mathematics from Al-Farabi Kazakh National University in 2002 and her degree candidate of technical sciences (Almaty, Kazakhstan) in 2009. Currently she is a leading researcher in the laboratory "Information Security" at the Institute of Information and Computing Technology and an associate professor at the Department of "Information systems" at Al-Farabi Kazakh National University. Area of scientific work: development and research in the field of information protection. She can be contacted at email: nkapalova@mail.ru.

Kunbolat Algazy received a master's degree in mathematics from Al-Farabi Kazakh National University in 2001 and a Ph.D. degree in information security systems, Almaty, Kazakhstan, in 2021. In 2001-2014 he worked in the field of information protection in the state structure. Between 2014 and 2016, he worked as a teacher at the Department of Mathematics at Satbayev University. Currently, he is a researcher in the laboratory "Information security" at the Institute of Information and Computing Technology. His research interests include cryptography, cryptanalysis, development, and research in the field of information protection. He can be contacted at email: kunbolat@mail.ru.

Armanbek Haumen received a master's degree in mathematics from Al-Farabi Kazakh National University in 2002. In 2000-2005, he worked in the field of information technology in the banking structure. In 2005-2014 he worked as a teacher in secondary schools. From 2014 to the present, he has been a researcher at the Information Security Laboratory of the Institute of Information and Computing Technologies. His research interests include cryptography, cryptanalysis, development, and research in the field of information security. He can be contacted at email: haumen.armanbek@gmail.com.

Kairat Sakan graduated from the Faculty of Mechanics and Mathematics of the Al-Farabi Kazakh National University, majoring in "mathematics and applied mathematics" (KazNU, Almaty, Kazakhstan) in 2001. In 2001-2002, he worked as a teacher at the Department of Applied Mathematics and Mathematical Modeling at KazNU. Between 2003 and 2005, he worked as a junior researcher at the Research Institute of Mathematics and Mechanics (IMM) of KazNU. After that, he worked in the field of information protection in the state structure for several years. Since 2018, he has been working as a mathematician in the information protection laboratory at the Scientific Institute of Information and Computing Technologies. Currently, he is a doctoral student at KazNU, majoring in information security systems. The field of scientific research is information protection in the public and private sector. He can be contacted at email: 19kairat78@gmail.com.