NTU JOURNAL OF PURE SCIENCES
EISSN: 2789-1097

RESEARCH ARTICLE
Open Access

# A Comparison of the Randomness Analysis of the Modified RECTANGLE Block Cipher Algorithm and Original Algorithm

Shatha A. Baker [1] Dr. Ahmed S. Nori[2]

[1]Ph.D. student in Mosul University / Iraq, shathaab@ntu.edu.iq
[2]Assistant Professor in Mosul University/ Iraq , Ahmed.s.nori@uomosul.edu.iq

**Received:** 04-02-2022, **Accepted: 09-03-2022** , **Published online : 24-05-2022**

**Abstract.** In recent years, different encryption lightweight algorithms have been suggested to protect the security of data transferred across the IoT network. The symmetric key ciphers play a significant role in the security of devices, in particular block ciphers, the RECTANGLE algorithm amongst the current lightweight algorithms. RECTANGLE algorithm does have good encryption efficacy but the algorithm lacks the diffusion and confusion properties that a block cipher should provide as one of its cryptographic security aspects. Therefore, to improve the diffusion and confusion properties of the algorithm, we expanded RECTANGLE utilizing a 3D cipher and modified the key scheduling algorithm. To assess the randomness of the algorithm, randomness analysis was done by using the NIST Statistical Test Suite. To create 100 samples, nine distinct data categories were used. The algorithm created ciphertext blocks, which were then concatenated to form a binary sequence. NIST tests carried out less than 1% significance level. According to the results of the comparison study, the proposed algorithm's randomness analysis results are gave 27.49% better results than the original algorithm.

**Keywords:** Security, RECTANGLE algorithm, Randomness Analysis, Data Categories, NIST.

## Introduction

RECTANGLE is a lightweight block cipher that allows different platforms to be implemented fast. It needs to adopt a 25-round (SP) structure. It has a 64-bit block size and supports 80 or 128-bit keys. In both software and hardware environment, RECTANGLE provides high efficiency, that offers ample flexibility for various application scenarios[1]. But the algorithm lacks the property of confusion which an algorithm can give. The weakest point of the algorithm seems to be its non-robust generation of round keys[2]. The key scheduling algorithm is a significant factor that has a major impact on the security of a cryptographic algorithm. A strong key scheduling algorithm can generate round keys with independent, random and not associated properties, with a statistical analysis that can be validated[3].

An expansion of the RECTANGLE algorithm was proposed in this paper utilizing the 3D cipher architecture and modifying the algorithm for the key schedule to enhance the algorithm’s diffusion and confusion without growing block and key sizes. This algorithm is very effective for the resource-restricted devices of the IoT and the prime aim of the improvement is to improve the algorithm's security power.

The randomness test refers to the strategies that were considered while assessing the minimum security requirement for a cryptographic algorithm. Statistical analysis may determine whether an algorithm meets the security criteria or not[4]. The Rectangle algorithm was subjected to statistical analysis, based on a 1% significance level; the results showed that it is not random. Therefore, to enhance the results of randomness, the proposed algorithm was presented, which improved the results, and this was shown by the results of the statistical analysis that was conducted on it.

The rest of the paper is arranged as below. In section 2, related work is addressed followed by a RECTANGLE's specification which is described in section 3. A description of the algorithm proposed is given in section 4. Section 5 introduces the randomness test while the section 6 presents the results and analysis eventually the conclusion is set out in Section 7.

## Related Works

Multiple researchers have suggested metrics for assessing the features of traditional and lightweight encryption algorithms. One of these metrics is randomness analysis such as, in 2020, the authors in[5] presented a randomness analysis of the Rectangle algorithm using 1000 samples and based on (80-bit) and (120-bit) keys. The results showed that the algorithm with 80-bit key passed 13 of 15 tests, while the algorithm with 120-bit key was able to pass 11 of 15 tests. Therefore, they suggested that modifications should be made to the algorithm to improve its security. The paper[6] in 2020, examined the PRESENT algorithm abilities as generator of random number. The PRESENT algorithm is subjected to a randomness examination using the NIST test. Six categories of data were applied to create 100 samples. However, the outcomes provided by the analysis non-random are depending on the 1% significance level. To strengthen security, it is suggested that the PRESENT be enhanced in the future. In 2020, The paper[7] suggested an extension using 3D bit rotation to the RECTANGLE algorithm. Without increasing block and key sizes, this optimization raised the randomness test value by 22.22%. The success rate implies that the ciphertext's non-random features have been decreased.

In 2018, the Ref.[8] provides a comprehensive analysis of the randomness of different algorithms Kasumi, Simeck, AES and DES. Depending on the NIST statistical test, the analysis was performed on five measures. The evaluation results showed that the algorithms have different levels of randomness. While in Ref.[9] in 2011, presented an analysis of the NIST randomness test of the KATAN cipher algorithm and used 100 samples based on Nine data sets and 1% significance level. They concluded that the algorithm could not pass all tests of randomness.

## RECTANGLE Block Cipher

` RECTANGLE block cipher is a lightweight algorithm operates on 64-bit block size developed by[10] . It can support either 80 or 128 bits of a key. RECTANGLE performs 25 round SP network, each round has three stages: Add

Round Key, SubColumns. ShiftRows, as shown in figure 1.

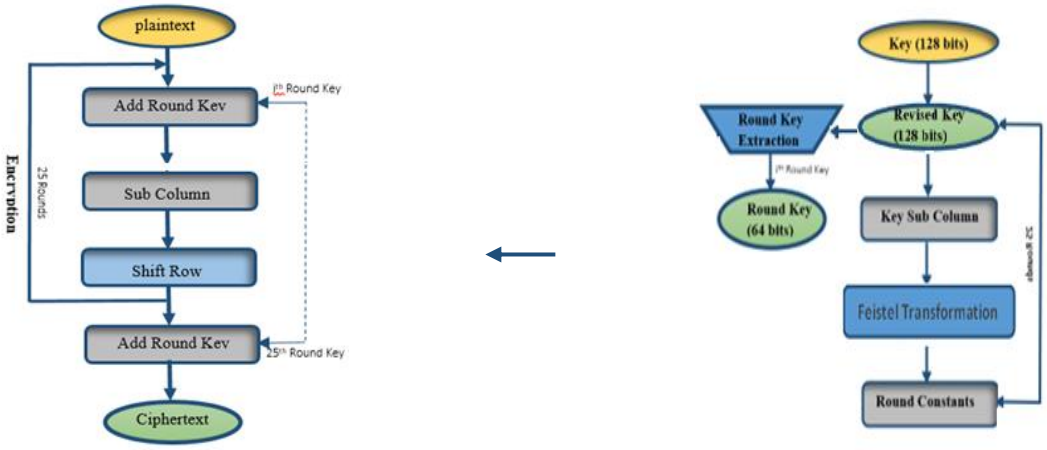


**Figure1**. RECTANGLE Process

**1 Encryption Algorithm**

A plaintext of 64 bits, called a cipher state. It for the encryption process is split into four 16-bit parts and is represented in the format RECTANGLE 4 x 16, which is the source of the RECTANGLE cipher name. Three steps are carried out in 25 rounds of RECTANGLE[11]:

**1.1 Add Round Key:**

A bitwise XOR of the round subkey to the cipher state, the round subkey is changed for each round by using key scheduling.

**1.2 Column Substitution:**

Parallel application of S-boxes shows to the 4 bits in the same column. The S-box used for getting the result is provided in hexadecimal notations in Table1.

The S-box input is $\mathrm{Col}(j) = a_{3,j}||a_{2,j}||a_{1,j}||a_{0,j}$ for $0 \leqslant j \leqslant 15$

and the output is $S(\mathrm{Col}(j)) = b_{3,j}||b_{2,j}||b_{1,j}||b_{0,j}$

**Table 1:** S-Box

| x | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| S(x) | 6 | 5 | c | a | 1 | E | 7 | 9 |
| x | 8 | 9 | a | b | c | d | e | f |
| S(x) | b | 0 | 3 | d | 8 | f | 4 | 2 |

**1.3 Shift Row:**

The 64-bit cipher state is split into four rows of 16-bit rows. A left rotation to each row across will be various offsets. This rotation will occur as shown below:

- $\mathrm{Row'}_0 := \mathrm{Row}_0 <<< 0$
- $\mathrm{Row'}_1 := \mathrm{Row}_1 <<< 1$
- $\mathrm{Row'}_2 := \mathrm{Row}_2 <<< 12$
- $\mathrm{Row'}_3 := \mathrm{Row}_3 <<< 13$

**2 Key scheduling**

RECTANGLE supports two, 80-bit and 128-bit key sizes. Column Substitution, Feistel Transformation and Round Counter XOR operations are identical to the main 80-bit and 128-bit key scheduling algorithm. For proposed implementations of architectures, the 128-bit key scheduling algorithm is utilized in this paper. Let $V = v_{127} \ldots v_1 v_0$ describes a key is kept in the key register and organized in 4 × 32 to rely on 4-bit S-boxes[12].

To generate the 64-bit of the $i^{th}$ subkey $K_i$ at round, the 16 rightmost columns of each RowKey are taken as a round key. After the round key has been extracted, the key register is modified in three phases[13].

**a. Column substitution**: implement the S-box to the 8 rightmost columns, i.e.,

$$k'_{3,j}\|k'_{2,j}\|k'_{1,j}\|k'_{0,j} := S\left(k_{3,j}\|k_{2,j}\|k_{1,j}\|k_{0,j}\right), \qquad 0 \leq j \leq 7$$

**b. Feistel transformation:** The Feistel transformation is implemented to each round

- $\mathrm{RowKey'}_0 := (\mathrm{RowKey}_0 <<< 8) \oplus \mathrm{RowKey}_1$
- $\mathrm{RowKey'}_1 := \mathrm{RowKey}_2$
- $\mathrm{RowKey'}_2 := (\mathrm{RowKey}_2 <<< 16) \oplus \mathrm{RowKey}_3$
- $\mathrm{RowKey'}_3 := \mathrm{RowKey}_0$

**c. Add Round Constant:** The 5-bit key state (κ0,4||κ0,3||κ0,2||κ0,1||κ0,0) XORed with 5-bit constant RC[i].

$$k'_{0,4}||k'_{0,3}||k'_{0,2}||k'_{0,1}||k'_{0,0} := (k_{0,4}||k_{0,3}||k_{0,2}||k_{0,1}||k_{0,0}) \oplus RC[i]$$

The RC's initial value is the value of (0x 1). The 5 bits of RC (rc4, rc3, rc2, rc1, rc0) are left to be shifted by 1 in each round, as shown in table2.

**Table2.** Round Constants

| i | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| **Rc[i]** | 01 | 02 | 04 | 09 | 12 | 05 | 0B | 16 | 0C |
| **i** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** |
| **Rc[i]** | 19 | 13 | 07 | 0F | 1F | 1E | 1C | 18 | 11 |
| **i** | **18** | **19** | **20** | **21** | **22** | **23** | **24** | | |
| **Rc[i]** | 03 | 06 | 0D | 1B | 17 | 0E | 1D | | |

## Proposed Algorithm

The proposed rectangle expansion algorithm includes two parts, the first is a three-dimensional bit rotation, and the second is a modification of the key schedule algorithm.

### 1 3D Bit Rotation

A cube is a plaintext array carried out on a 3dimensional (3D) bit array. The 3D cipher architecture introduces diffusion and confusion that enhances the algorithm. Because the original RECTANGLE's cipher state is split into four rows of plaintext, by putting 4 x 4 x 4 matrices with the input data bits, it is possible to convert the state into a 3D state that demands 4 plaintext slices, as presented in Figure 2.

Let $W = w_{63}k...kw_1kw_0$ signify the cipher state. A $w_{15}k...kw_1kw_0$ are the initial 16 bits, are placed in $Slice_0$ and $Slice_1$ contain the next 16 bits $w_{31}k...kw_{17}kw_{16}$. While the $Slice_2$ and $Slice_3$ contains the consecutive 16 bits of the cipher states, as shown in Figure 2.

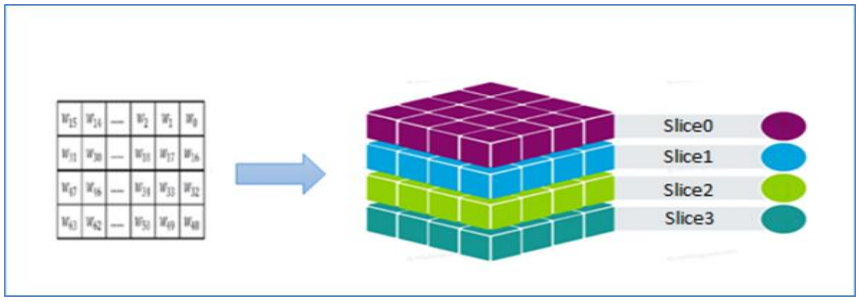


**Figure 2:** 3D Cipher state

In a specific clockwise direction, 4 slices are rotated

- $Slice_0$=Rotate 0°
- $Slice_1$=Rotate 90°
- $Slice_2$=Rotate 180°
- $Slice_3$= Rotate 270°

The suggested 3D Bit Rotation provides the original algorithm with greater confusion and diffusion. It added to each round in algorithm, as seen in Figure 3.

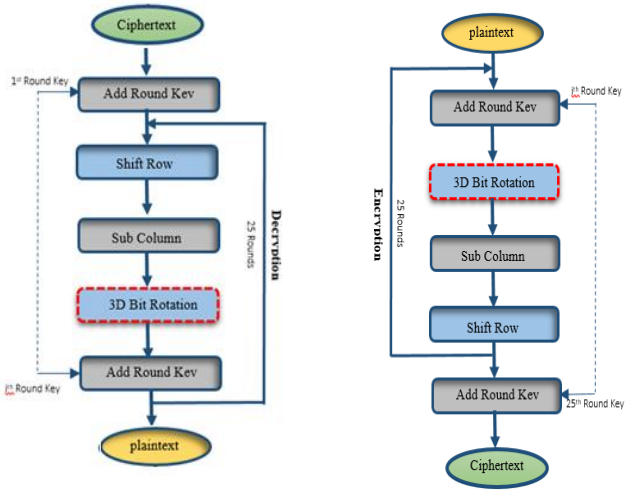


**Figure3.** Proposed Process

**2 Modifications of Key Schedule**

There are two steps to the improvements proposed to the key schedule algorithm:

- The Feistel transformation process is slightly changed.

$$\begin{aligned} \text{Row}_0 &\rightarrow (\text{Row}_2 \lll 16) \oplus \text{Row}_3 \\ \text{Row}_1 &\rightarrow \text{Row}_2 \\ \text{Row}_2 &\rightarrow (\text{Row}_0 \lll 8) \oplus \text{Row}_1 \\ \text{Row}_3 &\rightarrow \text{Row}_0 \end{aligned}$$

- Rather than 16 columns to the right of the current key register, a round key is extracted by the first two rows.

## Randomness Test

The test is a statistical package consisted of 15 tests constructed to evaluate the degree of randomness generated by block cipher algorithms for binary sequences. Therefore, the randomness strength test of the proposed algorithm is tested using the NIST-defined statistical test[14], table 3 shows the tests and length (n) of sequence bits for each test .

The p-value is computed by these tests, the significance level (α) should be specified in order to determining the ciphertext's randomness, the α has been set to 1% (0.01)[5].

If the p-value ≥ α, the ciphertext is assumed to be random, on the other hand, if the p-value < α, the ciphertext is regarded as non-random[15].

The following formula was used to calculate the maximum number of failed samples accepted:

$$\gamma = s(\alpha + 3\sqrt{(\alpha(1-\alpha))/s})$$

Where γ =number of failed samples acceptable, α is the significance level which equals to 0.01 and s represents the sample size of 100 ciphertexts.
If the rejection number goes above the proportion pα, then the sample is non-random[16].
Depending on the formula, any test will produce a p_value; the rate of rejection must not exceed three sequences.

**Table 3:** Statistical Tests Description[17]

| Statistical Test | Test purpose | Recommended bits | P_value |
|---|---|---|---|
| Frequency Test | To determine the number of ones and zeros in a sequence. | n ≥ 100 | 1 |
| Run Test | To determine the number of runs of ones and zeros of various lengths. | n ≥ 100 | 1 |
| Longest Run of One's Test | To determine the distribution of long runs of ones. | n ≥ 750,000 | 1 |
| Spectral (Discrete Fourier Transform) Test | To determine the spectral frequency of the binary sequence. | n ≥ 1,000 | 1 |
| Block Frequency Test | To determine the frequency of m-bit blocks in a sequence. | n ≥ 100 | 1 |
| Binary Matrix Rank Test | To determine the distribution of the rank of 32 × 32 bit matrices. | n ≥ 38,912 | 1 |
| Cumulative Sums (Forward/Reverse) | To assess if the number of partial sequences occurring in the sequence being checked is either too big or too small. | n ≥ 100 | 2 |
| Random Excursion Test | To examine the number of cycles within a sequence. | n ≥ 1,000,000 | 8 |
| Random Excursion Variant Test | To detect the difference between the distribution of the number of visits in a random walk and that in a given state. | n ≥ 1,000,000 | 16 |
| Non-overlapping Template Test | To determine the number of occurrences for a specified non periodic template. | Not specified | 148 |
| Overlapping Template Matching Test | To determine the number of occurrences for a template of all ones. | n ≥ 1,000,000 | 1 |
| Universal Statistical Test | To determine a binary sequence does not compress beyond what is expected of a truly random sequence. | n ≥ 387,480 | 1 |
| Linear Complexity Test | The purpose of this test is to determine whether or not the sequence is complex enough to be considered truly random. | n ≥ 1,000,000 | 1 |
| Serial Test | To decide whether the number of occurrences of m-bit overlapping patterns is essentially the same as that expected in a random sequence. (m-bit is the length of bits for each block). | Not specified | 2 |
| Approximate Entropy Test | To compare the frequency of overlapping blocks of two consecutive/adjacent lengths | Not specified | 1 |

The outcome is passed if the number of rejected sequences is fewer than or equal to the rate of rejection. Otherwise, the outcome is fail.

The tests are divided into two groups: parameterized tests and non-parameterized tests. The parameterized test requires the number of blocks; block size and template length in order to get the p-values for each test, whereas we don't need any additional parameters in the non-parameterized test

**1 Data Categories for algorithms**

A block cipher generates ciphertext that includes the sequence of bits with the block size. However, in order to assess an algorithm's randomness, a big bit sequence must be present in the input data. Nine categories of data are utilized to build data input in the form of plaintext and key data. Table 4 provide the number of bits required for each data category[16].

### 1.1 Strict Key Avalanche (SKA)

The purpose of SKA is to examine the algorithm's sensitivity to key changes. 100 samples were created. The plaintext is made up of zeros and the base key, a 128-bit random key is created. To obtain the base ciphertext, encrypt a plaintext with the base key. The base key is then flipped at the ith bit, for $1 \leq i \leq 128$. Then, each perturbed key is used to encrypt the plaintext to produce a derived block which is XORed with the base ciphertext before being concatenated into a long bit sequence. This process is repeated 123 times to get a sequence length of (1,007,616 bits).

### 1.2 Strict Plaintext Avalanche (SPA)

The purpose of SPA is to examine the algorithm's sensitivity to plaintext changes. 100 samples were created. The key is made up of

zero and the base plaintext consists from a 64-bit random plaintext. To obtain the base ciphertext, encrypt a base plaintext with the key. The base plaintext is then flipped at the ith bit, for $1 \leq i \leq 64$. Then, each perturbed plaintext is encrypted with the key to produce a derived block which is XORed with the base ciphertext before being concatenated into a long bit sequence. This process is repeated 245 times to get a sequence length of (1,003,520 bits).

### Plaintext/Ciphertext Correlation (PCC)

### 1.3

The purpose of this data categories is to check the correlation between each pair of (plaintext / ciphertext) by using ECB mode for every sample. The sequence use 15,625 blocks of random plaintext and one random key. To create a derived block, each random plaintext is encrypted with the same key then an XOR operation is applied between the plaintext and the corresponding ciphertext and following that they merged to make (1,000,000bit).

### 1.4 Ciphertext Block Chaining Mode (CBCM)

The CBCM is generated by utilizing the CBC mode. Each sample uses plaintext is made up of zeros, a random key and an initialization vector (IV) is made up of zeros. The initial ciphertext block ($CT_1$) is computed as

$$CT_1 = E_k(IV \oplus PT_1)$$

while subsequent ciphertext blocks is computed as

**Table 4:** Data Category Requirements

| No. | Data Category | RECTANGLE-128 | | | |
|---|---|---|---|---|---|
| | | Key | Plaintext | Derived Blocks | Derived Bits |
| 1. | Strict Key Avalanche (SKA) To inspect the sensitiveness of block ciphers to the key bits modifications. | 123 random 128-bit keys | All zero | 15,744 | 1,007,616 |
| 2. | Strict Plaintext Avalanche (SPA) To inspect the sensitiveness of block ciphers to the plaintext bit modifications. | All zero | 245 random 64-bit plaintext | 15,680 | 1,003,520 |
| 3. | Plaintext/Ciphertext Correlation (PCC) To inspect the relation between plaintext and ciphertext pairs using ECB mode of operation. | 1 random 128-bit key | 15,625 random 64-bit plainext | 15,625 | 1,000,000 |
| 4. | Ciphertext Block Chaining Mode (CBCM) To inspect the randomness of ciphertext using the CBC mode of operation. | 1 random 128-bit key | All zero | 15,625 | 1,000,000 |
| 5. | Random Plaintext/Random Key (RPRK) To inspect the randomness of ciphertext using random plaintext and random key. | 1 random 128-bit key | 15,625 random 64-bit plaintext | 15,625 | 1,000,000 |
| 6. | Low-Density Key (LDK) To inspect the randomness of ciphertext on the basis of low-density keys. | 3,241 specific 128-bit keys | 8,257 random 64-bit plaintext | 8,257 | 528,448 |
| 7. | High-Density Key (HDK) To inspect the randomness of ciphertext on the basis of high-density keys. | 3,241 specific 128-bit keys | 8,257 random 64-bit plaintext | 8,257 | 528,448 |
| 8. | Low-Density Plaintext (LDP) To inspect the randomness of ciphertext on the basis of low-density plaintext. | 2,081 specific 128-bit keys | 2,081 random 64-bit plaintext | 2,081 | 133,184 |
| 9. | High-Density Plaintext (HDP) To inspect the randomness of ciphertext on the basis of high-density plaintext. | 2,081 specific 128-bit keys | 2,081 random 64-bit plaintext | 2,081 | 133,184 |

$$CT_{i+1} = E_k\left(CT_i \oplus PT_i\right) \text{ for } 1 \leq i \leq 15{,}625.$$

To generate a large bit sequence (1,000,000 bit), produced ciphertext blocks are combined.

### 1.5 Random Plaintext / Random Key (RPRK)

The RPRK is utilized to check ciphertext randomness using random plaintext and random key. Each sample utilizes 15,625 blocks of random plaintext (64-bit) and one random key (128-bit). Derived blocks are computed in ECB mode that will be combined to generate a large sequence (1,000,000 bit).

### 1.6 Low Density Key (LDK)

LDK is generated depend on low density (x-bit) keys and utilizing ECB mode. The sample uses random plaintext and specific key. The first plaintext (64-bit) block is encrypted with key (128-bit) made up of zeros. Then, a key that has one at each bit position with the other bits set to zeros is used to encrypt the plaintext block. Following that, to encrypt the plaintext block, a key with two ones in every combination of two bit places and other bits set to zeros is employed. The total derived blocks are $1+C_1^{128}+C_2^{128}= 8{,}257$, as seen in Fig. 4. Each sequence includes 8,257×64bit =528,448 bit. The process is repeated to obtain 100 sequences.

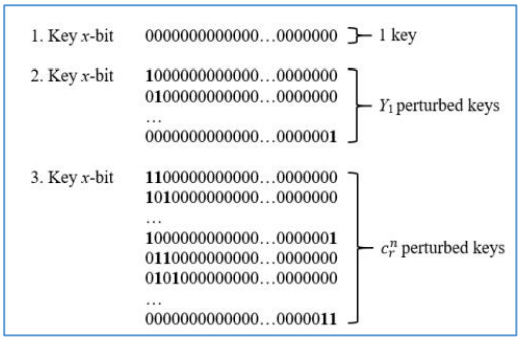


**Figure 4**: LDK data category

### 1.7 High Density Key (HDK)

HDK is generated depend on high density (x-bit) keys and utilizing ECB mode. The sample uses random plaintext and specific key. The first plaintext (64-bit) block is encrypted with key (128-bit) made up of ones. Then, a key that has zero at each bit position with the other bits set to ones is used to encrypt the plaintext block. Following that, to encrypt the plaintext block, a key with two zeros in every combination of two bit places and the rest bits set to ones is employed. The total derived blocks are $1+C_1^{128}+C_2^{128}=8,257$. Each sequence includes 8,257×64bit = 528,448bit. The process is repeated to obtain 100 sequences.

### 1.8 Low Density Plaintext (LDP)

LDP is generated depend on low density (y-bit) plaintexts and utilizing ECB mode. The sample uses random key and specific plaintext. The first plaintext (64bit) block made up of zeros is encrypted with a key (128bit). Then, a plaintext that has one at each bit position with the other bits set to zeros is used to encrypt with key. Following that, a plaintext with two ones in every combination of two bit places and other bits set to zeros is employed to encrypt with key. The total derived blocks are $1+C_1^{64}+C_2^{64}=2,081$. Each sequence includes 2,081×64bit=133,184 bit. The process is repeated to obtain 100 sequence

### 1.9 High Density Plaintext (HDP)

HDP is generated depend on high density (y-bit) plaintexts and utilizing ECB mode. The sample uses random key and specific plaintext. The first plaintext (64bit) block made up of ones is encrypted with a key (128bit). Then, a plaintext that has zero at each bit position with the other bits set to ones is used to encrypt with key. Following that, a plaintext with two zeros in every combination of two bit places and other bits set to ones is employed to encrypt with key. The total derived blocks are $1+C_1^{64}+C_2^{64}=2,081$. Each sequence includes 2,081×64bit=133,184 bit. The process is repeated to obtain 100 sequences.

## Results and Analysis

There are several steps to the process of assessing randomness analysis, i.e., taking a

sample utilizing data categories and applying the algorithm, running statistical randomness checks, and assessing the results.

SKA, PCC, SPA, RPRK and CBCM, were able to pass all 15 tests; while LDK, HDK, HDP and LDP, were able to pass 11 tests. Tables 5 demonstrate and compare analysis result between the RECTANGLE algorithm [7] and the proposed algorithm. As a result, during the experimentation, each sample will generate 1,578 P-values. There are a total of 1578 P-values examined because there is algorithm with 100 samples each. A 900 binary sequences were examined during the experiment (i.e. 100 samples x 9 categories of data).

**Table 5:** Randomness analysis result

| | | Parameterized Test Selection | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Block Frequency | | Non-Overlapping | | Overlapping | | Maurer's Universal | |
| | | Rectangle algorithm | Proposed algorithm | Rectangle algorithm | Proposed algorithm | Rectangle algorithm | Proposed algorithm | Rectangle algorithm | Proposed algorithm |
| Data Category | SKA | 0 | 0 | 19 | 8 | 0 | 0 | 0 | 0 |
| | SPA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | PCC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | CBCM | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| | RPRK | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| | LDK | 0 | 0 | 0 | 0 | N/A | | 0 | 0 |
| | HDK | 0 | 0 | 0 | 0 | | | 0 | 0 |
| | LDP | 0 | 0 | 0 | 0 | | | 0 | 0 |
| | HDP | 0 | 0 | 0 | 0 | | | 0 | 0 |

| | | Linear Complexity | | Serial | | Approximate Entropy | |
|---|---|---|---|---|---|---|---|
| | | Rectangle algorithm | Proposed algorithm | Rectangle algorithm | Proposed algorithm | Rectangle algorithm | Proposed algorithm |
| Data Category | SKA | 0 | 0 | 0 | 0 | 0 | 0 |
| | SPA | 0 | 0 | 0 | 0 | 0 | 0 |
| | PCC | 0 | 0 | 0 | 0 | 0 | 0 |
| | CBCM | 0 | 0 | 0 | 0 | 0 | 0 |
| | RPRK | 0 | 0 | 0 | 0 | 0 | 0 |
| | LDK | N/A | | 0 | 0 | 0 | 0 |
| | HDK | | | 0 | 0 | 0 | 0 |
| | LDP | | | 0 | 0 | 0 | 0 |
| | HDP | | | 0 | 0 | 0 | 0 |

| | | Non-Parameterized Test Selection | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Frequency | | Runs | | Longest Runs of Ones | | Spectral DFT | |
| | | Rectangle algorithm | Proposed algorithm | Rectangle algorithm | Proposed algorithm | Rectangle algorithm | Proposed algorithm | Rectangle algorithm | Proposed algorithm |
| Data Category | SKA | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| | SPA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | PCC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | CBCM | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| | RPRK | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | LDK | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | HDK | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | LDP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | HDP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | Binary Matrix Rank | | Cumulative Sums | | Random Excursion | | Random Excursion Variant | |
|---|---|---|---|---|---|---|---|---|---|
| | | Rectangle algorithm | Proposed algorithm | Rectangle algorithm | Proposed algorithm | Rectangle algorithm | Proposed algorithm | Rectangle algorithm | Proposed algorithm |
| Data Category | SKA | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 |
| | SPA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | PCC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | CBCM | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | RPRK | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | LDK | 0 | 0 | 0 | 0 | N/A | | N/A | |
| | HDK | 0 | 0 | 0 | 0 | | | | |
| | LDP | 0 | 0 | 0 | 0 | | | | |
| | HDP | 0 | 0 | 0 | 0 | | | | |

7,500 test values were generated using five categories of data that tested all 15 tests (i.e. 100 samples x 5 categories of data x 15 tests), while 4,400 test values were generated using four categories of data that tested only 11 tests (i.e. 100 samples x 4 categories of data x 11 tests), As a result, 11,900 test values were generated.

Both algorithms are incapable of passing all statistical tests. as shown by the results. The RECTANGLE method passed 11 of 15 statistical tests, but failed the tests for Longest Runs of Ones, Frequency, Non-Overlapping Templates and Cumulative Sums tests. Meanwhile, the proposed algorithm passed 14 of 15 statistical tests and failed the Non-Overlapping Templates test only. RECTANGLE algorithm passed (66.67%) of statistical tests, while the proposed algorithm passed (94.16%) statistical tests. In general, the proposed algorithm boosted the original algorithm's randomization.

## Conclusion

In this study, a randomization analysis was offered using the NIST test of the proposed algorithm and then it compared with the original algorithm. 100 samples were performed using nine data categories and the level of significance was chosen at 1%, to assess the algorithm under test is random or not.

The results show that the proposed algorithm is 27.49% better than original algorithm. The modification made to the original algorithm improved the randomness results. In general, the proposed algorithm was able to reduce the non-random properties of the ciphertext.

## References

[1] A. Aghaie, M. M. Kermani, and R. Azarderakhsh, “Fault diagnosis schemes for secure lightweight cryptographic block cipher RECTANGLE benchmarked on FPGA,” in *2016 IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2016, pp. 768–771.

[2] W. Zhang, Z. Bao, V. Rijmen, and M. Liu, “A New Classification of 4-bit Optimal S-boxes and Its Application to PRESENT, RECTANGLE and SPONGENT,” in *International Workshop on Fast Software Encryption*, 2015, pp. 494–515.

[3] M. Rana, Q. Mamun, and R. Islam, “Current Lightweight Cryptography Protocols in Smart City IoT Networks: A Survey,” pp. 1–22, 2020.

[4] S. Ariffin and N. A. M. Yusof, “Randomness analysis on 3D-AES block cipher,” in *2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, 2017, pp. 331–335.

[5] A. A. Zakaria, A. H. Azni, F. Ridzuan, N. H. Zakaria, and M. Daud, “Randomness analysis on RECTANGLE block cipher,” in *Cryptol. Inf. Secur. Conf*, 2020, pp. 133–142.

[6] L. M. Shamala, G. Zayaraz, K. Vivekanandan, and V. Vijayalakshmi, “Lightweight cryptography algorithms for internet of things enabled networks: An overview,” *J. Phys. Conf. Ser.*, vol. 1717, no. 1, 2021.

[7] A. A. Zakaria, A. H. Azni, F. Ridzuan, N. H. Zakaria, and M. Daud, “Extended RECTANGLE Algorithm Using 3D Bit Rotation to Propose a New Lightweight Block Cipher for IoT,” *IEEE Access*, vol. 8, pp. 198646–198658, 2020.

[8] M. Qasaimeh, R. S. Al-Qassas, and S. Tedmori, “Software randomness analysis and evaluation of lightweight ciphers: the prospective for IoT security,” *Multimed. Tools Appl.*, vol. 77, no. 14, pp. 18415–18449, 2018.

[9] N. H. Lot, N. A. N. Abdullah, and H. A. Rani, “Statistical analysis on KATAN block cipher,” in *2011 International Conference on Research and Innovation in Information Systems*, 2011, pp. 1–6.

[10] W. Zhang, Z. Bao, D. Lin, V. Rijmen, B. Yang, and I. Verbauwhede, “RECTANGLE: a bit-slice lightweight block cipher suitable for multiple platforms,” *Sci. China Inf. Sci.*, vol. 58, no. 12, pp. 1–15, 2015.

[11] A. A. Zakaria, A. H. Azni, F. Ridzuan, N. H. Zakaria, and M. Daud, “Modifications of Key Schedule Algorithm on RECTANGLE Block Cipher,” in *International Conference on Advances in Cyber Security*, 2020, pp. 194–206.

[12] M. A. Philip, V. Vaithiyanathan, and K. Jain, “Implementation analysis of rectangle cipher and its variant,” in *2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, 2018, pp. 474–479.

[13] V. Dahiphale, H. Raut, and G. Bansod, “Design and Implementation of novel datapath designs of lightweight cipher RECTANGLE for resource constrained environment,” *Multimed. Tools Appl.*, vol. 78, no. 16, pp. 23659–23688, 2019.

[14] R. S. Villafuerte, A. M. Sison, A. A. Hernandez, and R. P. Medina, “Randomness Evaluation of the Improved 3D-Playfair (i3D) Cipher Algorithm,” in *2020 12th International Conference on Communication Software and Networks (ICCSN)*, 2020, pp. 240–245.

[15] S. Zhu, Y. Ma, J. Lin, J. Zhuang, and J. Jing, “More powerful and reliable second-level statistical randomness tests for NIST SP 800-22,” in *International Conference on the Theory and Application of Cryptology and Information Security*, 2016, pp. 307–329.

[16] I. N. M. Shah, H. A. Rani, M. M. Ahmad, and E. S. Ismail, “Cryptographic Randomness Analysis on Simon32/64.”

[17] L. C. N. Chew, I. N. M. Shah, N. A. N. Abdullah, N. H. A. Zawawi, H. A. Rani, and A. A. Zakaria, “Randomness analysis on speck family of lightweight block cipher,” *Conf. Proc. - Cryptol. 2014 Proc. 4th Int. Cryptol. Inf. Secur. Conf. 2014*, vol. 5, no. June 2013.