International Journal of Cryptology Research 10(2): 1 - 22 (2020)

# Randomness Tests on Nine Data Categories of RECTANGLE Using NIST Statistical Test Suite

**Abdul Alif Zakaria**[*1,3], **A. H. Azni**[*2], **Farida Ridzuan**[2], **Nur Hafiza Zakaria**[1], and **Maslina Daud**[3]

[1]*Faculty of Science and Technology, Universiti Sains Islam Malaysia, Negeri Sembilan 71800, Malaysia*
[2]*CyberSecurity and System Research Unit, Islamic Science Institute (ISI), Universiti Sains Islam Malaysia, Negeri Sembilan 71800, Malaysia*
[3]*CyberSecurity Malaysia, Selangor 63000, Malaysia*

*E-mail: alif@cybersecurity.my; ahazni@usim.edu.my;*
[*]*Corresponding author*

## ABSTRACT

RECTANGLE lightweight algorithm is a 64-bit block cipher using 80-bit and 128-bit key variants. A lightweight algorithm takes lesser computational power than a conventional algorithm. Implementing a lightweight algorithm in low-resource devices is more effective. To ensure the output has no pattern, randomness is an essential property for an algorithm. The NIST Statistical Test Suite is used to execute the randomness analysis. To produce 1,000 input sequences for each algorithm, nine data categories are implemented. RECTANGLE-80 and RECTANGLE-128 passed the randomness analysis with 98.73% and 98.48%. The results reveal that RECTANGLE appears to be non-random based on the 0.1% significance level. The analysis findings found weaknesses that can be explored in future research.

**Keywords:** Randomness, NIST statistical test suite, RECTANGLE, data category, lightweight block cipher

# 1 INTRODUCTION

Small computing devices such as smart cards, RFID tags, and node sensors have posed major security concerns (Khan and Salah (2018)). Given the security provided at a lower cost, lightweight block cipher gains much attention (Öğünç (2018)). Low energy consumption and high encryption speed are among the factors in the application of lightweight algorithms (Poschmann (2009)). Numerous algorithms have been proposed since 2011 such as Piccolo (Shibutani et al. (2011)), LED (Guo et al. (2011)), TWINE (Tomoyasu (2012)), SPARX (Dinu et al. (2016)), SIMON, and SPECK (Beaulieu et al. (2015)). In low-power and lossy networks, security is a huge concern, therefore more lightweight algorithms innovation is required.

RECTANGLE block cipher was designed for embedded devices (Zhang et al. (2015)). This cipher achieves low hardware costs and high software efficiency (Senol (2017)). Although RECTANGLE is efficient, more attention is required to its security. In order to increase its security, enhancements to RECTANGLE have been proposed (Zhang et al. (2015), Yan et al. (2019)). RECTANGLE could enhance its efficiency and security needed for embedded devices by analyzing its security.

Randomness is the minimum security criteria for a block cipher (Ariffin and Yusof (2017)). The randomness test will decide if the analyzed cipher satisfies the security requirement (Zakaria et. al. (2020)). A non-random algorithm is susceptible to cryptographic attacks (Isa and Z'aba (2012)). It should not be possible for an attacker to predict the cryptographic sequences with lesser effort than the brute force method (Chew et al. (2015)). Hence, a cryptographic algorithm should be able to generate random output. NIST Statistical Test Suite has been implemented to analyze many algorithms including AES, Serpent, Twofish, RC6, and MARS (Aljohani et al. (2019)). Therefore, RECTANGLE block cipher should be analyzed using the randomness test application.

The content of this analysis paper is structured as follows. The description of RECTANGLE block cipher is provided in Section 2. Section 3 defined the randomness tests and data category. The experimental results on RECTAN-

GLE are discussed in section 4. Section 5 summarizes the conclusion.

# 2 RECTANGLE BLOCK CIPHER

RECTANGLE comprises of 64-bit block size that supports 80-bit and 128-bit key identified as RECTANGLE-80 and RECTANGLE-128. This algorithm uses bit-slice methods with 25 encryption rounds (Tezcan et al. (2016)). RECTANGLE offers remarkable performance in hardware and software (Bao et al. (2015), Omrani et al. (2018)) that gives multiple application platforms flexibility.

## 2.1 Cipher and Subkey States

RECTANGLE cipher state is presented in 4 by 16 bits array (Feizi et al. (2015)). Let $W = w_{63}||\cdots||w_1||w_0$ represent the cipher state. In the first 16 bits, $w_{15}||\cdots||w_1||w_0$ are positioned in $Row(0)$ and the following 16 bits $w_{31}||\cdots||w_{17}||w_{16}$ are positioned in $Row(1)$ and so on. In each round, a 64-bit subkey is utilized as 4 by 16 array bits.

## 2.2 Encryption

RECTANGLE operates for 25 rounds using a substitution-permutation network. Three functions including $AddRoundKey$, $SubColumn$, and $ShiftRow$ are used in each round. After the last round, another $AddRoundKey$ function is implemented. The encryption process is outlined below:

1. *AddRoundKey*: Bitwise XOR operation of the cipher state ($a$) and the round subkey ($K$).
2. *SubColumn*: Column substitution implementing the RECTANGLE S-box. The S-box input is $Col(j) = a_{3,j}||a_{2,j}||a_{1,j}||a_{0,j}$ for $0 \leq j \leq 15$, and $S(Col(j)) = b_{3,j}||b_{2,j}||b_{1,j}||b_{0,j}$ represents the output.

3. *ShiftRow*: Row is left rotated in a specified position. $Row(0)$ remains constant while $Row(1)$, $Row(2)$, and $Row(3)$ are left rotated over 1, 12, and 13 bits respectively.

## 2.3 Key Schedule

RECTANGLE-80 is used as an illustration in this section. Let $V = v_{79}|| \cdots ||v_1||v_0$ represent the key input. 16 rightmost columns of the key are placed next to one another to construct the 64-bit $i^{th}$ subkey $K_i$ at round $i$. In every round, the key register are updated as listed below:

1. S-box rearranged $Column(0)$, i.e., $k_{3,0}||k_{2,0}||k_{1,0}||k_{0,0} = S(k_{3,0}||k_{2,0} ||k_{1,0}||k_{0,0})$.
2. Generalized Feistel transformation is applied, i.e., $Row(0) = Row(0) <<< 8 \oplus Row(1)$, $Row(1) = Row(2)$, $Row(2) = Row(3)$, $Row(3) = Row(3) <<< 12 \oplus Row(4)$, and $Row(4) = Row(0)$
3. 5-bit key state is XORed with the round constant $Rc[i]$, i.e., $(k_{4,0}||k_{3,0} ||k_{2,0}||k_{1,0}||k_{0,0}) = (k_{4,0}||k_{3,0}||k_{2,0}||k_{1,0}||k_{0,0}) \oplus Rc[i]$. Finally, from the revised key state $K_{25}$ is extracted.

# 3 RANDOMNESS TEST

The analysis is conducted on complete 25 encryption rounds of RECTANGLE using the NIST Statistical Suite that consists of 15 statistical tests with multiple input parameters (Rukhin et al. (2001)). The statistical test tool works on various ciphertext non-randomness characteristics. Eight tests are categorized as Non-Parameterized Test Selection that does not permit the user to input any parameter. On the other hand, seven tests are categorized as the Parameterized Test Selection that demands the user to enter parameter values. Table 1 listed each of the statistical tests and the $p$-values produced by each test.

| Test Selection | Statistical Test | $p$-value |
|---|---|---|
| Parameterized | Block Frequency | 1 |
| | Linear Complexity | |
| | Maurer's Universal | |
| | Approximate Entropy | |
| | Overlapping Templates | |
| | Serial | 2 |
| | Non-Overlapping | 148 |
| Non-Parameterized | Runs | 1 |
| | Frequency | |
| | Spectral DFT | |
| | Binary Matrix Rank | |
| | Longest Runs of Ones | |
| | Cumulative Sums | 2 |
| | Random Excursion | 8 |
| | Random Excursion Variant | 18 |

**Table 1:** Breakdown of the 188 $p$-values obtained by each sample.

Randomness tests require a significance level to be determined. The significance level must be selected from 0.1% to 1%, whereas the minimum sample should be at least the inverse of the significance level. The significance level, $\alpha$ has been set at 0.1% (0.001) for RECTANGLE, and the required number of samples for experiments is $1 \div 0.001 = 1,000$ samples. If the $p$-value $\geq \alpha$, the sample is random with 99.9% confidence level (Simion and Burciu (2019)). Meanwhile, the sample is not random if the $p$-value $< \alpha$.

In this analysis, the range of acceptable proportions for the ciphertext (Sýs et al. (2015)) is calculated using the confidence interval as defined in the following formula:

$$[p'_a, p'_b] = p' \pm 3\sqrt{\frac{p'(1-p')}{s}} \tag{1}$$

where $p' = 1 - \alpha$ , $\alpha$ is the significance level which equals to 0.001, and $s$ is the ciphertext sample size which equals to 1,000. The sample is considered not random if the proportion falls outside of the interval $[p'_a, p'_b]$ (Moussaoui et al. (2019)).

For a statistical test with one p-value, the acceptable rejection range is within 0 to 4 samples. Note that Serial and Cumulative Sums produce two $p$-values each that are independently analyzed. For Non-overlapping Template, although the test produces 148 $p$-values, they are individually analyzed. As such, the acceptable rejection range is also within 0 to 4.

Random Excursion and Random Excursion Variant tests may not make use of all 1,000 ciphertext. Several samples may not contain a sufficient number of cycles (500 cycles) needed for the tests. Therefore, the acceptable rejection ranges for both tests vary depending on the samples.

## 3.1 RECTANGLE Data Categories

Ciphertext produced from a block cipher contains the sequence of bits with the size of a block. However, to evaluate the randomness of an algorithm, the input data must contain a large bit sequence. Nine data categories are implemented in constructing the input plaintext and key data (Abdullah et al. (2011)) as illustrated in Table 2.

For every algorithm, 1,000 samples are produced using the data categories. The generated number of blocks for each sample depends on the key and block sizes (Abdullah et al. (2015)) of RECTANGLE-80 and RECTANGLE-128. To obtain a large bit sequence, the generated blocks are concatenated.

| No. | Data Category | RECTANGLE-80 | | | | RECTANGLE-128 | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Key | Plaintext | Derived Blocks | Derived Bits | Key | Plaintext | Derived Blocks | Derived Bits |
| 1. | Strict Key Avalanche (SKA)<br>To inspect the sensitiveness of block ciphers to the key bits modifications. | 196 random 80-bit keys | All zero | 15,680 | 1,003,520 | 123 random 128-bit keys | All zero | 15,744 | 1,007,616 |
| 2. | Strict Plaintext Avalanche (SPA)<br>To inspect the sensitiveness of block ciphers to the plaintext bit modifications. | All zero | 245 random 64-bit plaintext | 15,680 | 1,003,520 | All zero | 245 random 64-bit plaintext | 15,680 | 1,003,520 |
| 3. | Plaintext/Ciphertext Correlation (PCC)<br>To inspect the relation between plaintext and ciphertext pairs using ECB mode of operation. | 1 random 80-bit key | 15,625 random 64-bit plaintext | 15,625 | 1,000,000 | 1 random 128-bit key | 15,625 random 64-bit plainext | 15,625 | 1,000,000 |
| 4. | Ciphertext Block Chaining Mode (CBCM)<br>To inspect the randomness of ciphertext using the CBC mode of operation. | 1 random 80-bit key | All zero | 15,625 | 1,000,000 | 1 random 128-bit key | All zero | 15,625 | 1,000,000 |
| 5. | Random Plaintext/Random Key (RPRK)<br>To inspect the randomness of ciphertext using random plaintext and random key. | 1 random 80-bit key | 15,625 random 64-bit plaintext | 15,625 | 1,000,000 | 1 random 128-bit key | 15,625 random 64-bit plaintext | 15,625 | 1,000,000 |
| 6. | Low-Density Key (LDK)<br>To inspect the randomness of ciphertext on the basis of low-density keys. | 3,241 specific 80-bit keys | 3,241 random 64-bit plaintext | 3,241 | 207,424 | 3,241 specific 128-bit keys | 8,257 random 64-bit plaintext | 8,257 | 528,448 |
| 7. | High-Density Key (HDK)<br>To inspect the randomness of ciphertext on the basis of high-density keys. | 3,241 specific 80-bit keys | 3,241 random 64-bit plaintext | 3,241 | 207,424 | 3,241 specific 128-bit keys | 8,257 random 64-bit plaintext | 8,257 | 528,448 |
| 8. | Low-Density Plaintext (LDP)<br>To inspect the randomness of ciphertext on the basis of low-density plaintext. | 2,081 random 80-bit keys | 2,081 specific 64-bit plaintext | 2,081 | 133,184 | 2,081 specific 128-bit keys | 2,081 random 64-bit plaintext | 2,081 | 133,184 |
| 9. | High-Density Plaintext (HDP)<br>To inspect the randomness of ciphertext on the basis of high-density plaintext. | 2,081 random 80-bit keys | 2,081 specific 64-bit plaintext | 2,081 | 133,184 | 2,081 specific 128-bit keys | 2,081 random 64-bit plaintext | 2,081 | 133,184 |

**Table 2:** Data Categories

### 3.1.1 Strict Key Avalanche

Strict Key Avalanche (SKA) examines the sensitivity of an algorithm to modifications in the $x$-bit key. Every sample utilizes all-zeros plaintext and $X$ blocks of random $x$-bit base-keys as shown in Figure 1. For a fixed plaintext block, the avalanche effect is satisfied when any of the key bit is complemented and each ciphertext block changes with a probability of one half.

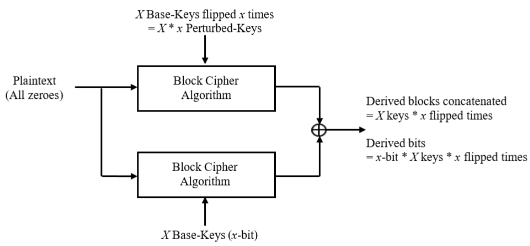


**Figure 1:** Strict Key Avalanche data category

For SKA, at least 1,000 sequences are required to be examined. First, a random 80-bit random key for RECTANGLE-80 (or 128-bit random key for RECTANGLE-128) is generated as the base-key. Encrypt an all-zeros plaintext with the base-key to get the base-ciphertext. Next, the base-key is flipped at the $i^{th}$ bit, for $1 \leq i \leq x$ giving a total of $X \times x$ perturbed-keys. The all-zeros plaintext is then encrypted with every perturbed-key. Each perturbed-ciphertext is XORed with the base-ciphertext then concatenated to obtain a large bit sequence.

To obtain a minimum of 1,000,000-bit ciphertext, the process is repeated 196 times for 80-bit random base-key of RECTANGLE-80 or 123 times for 128-bit random base-key of RECTANGLE-128. Each sequence contains 196 $\times$ 80-bit key $\times$ 64-bit block = 1,003,520-bit output for RECTANGLE-80 or 123 $\times$ 128-bit key $\times$ 64-bit block = 1,007,616-bit output for RECTANGLE-128. This process is repeated 1,000 times before executing the randomness test.

### 3.1.2 Strict Plaintext Avalanche

Strict Plaintext Avalanche (SPA) examines the sensitivity of an algorithm to modifications in the $y$-bit plaintext. Every sample utilizes all-zeros key and $Y$ blocks of random $y$-bit base-plaintext as shown in Figure 2. For a fixed key, the avalanche effect is satisfied when any plaintext bit is complemented and every ciphertext bit changes with a probability of one half.

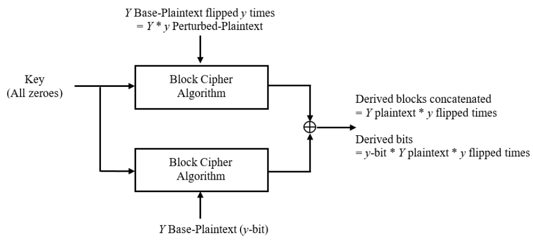


**Figure 2:** Strict Plaintext Avalanche data category

In SPA, the 80-bit key for RECTANGLE-80 (or 128-bit key for RECTANGLE-128) is fixed to be all-zeros. Then generate a random 64-bit plaintext as the base-plaintext and encrypt it using an all-zeros key to produce base-ciphertext. Next, each base-plaintext is flipped at the $i^{th}$ bit, for $1 \leq i \leq y$ giving a total of $Y \times y$ perturbed-plaintext. Each perturbed-plaintext is then encrypted using the all-zeros key. All resultant ciphertext of perturbed-plaintext is XORed with the ciphertext resulting from the encryption of its corresponding base-plaintext. The XOR output is called the derived block that will be concatenated to obtain a large bit sequence.

To obtain a minimum of 1,000,000-bit ciphertext, this process is to be repeated 245 times for RECTANGLE-80 or RECTANGLE-128. Every sequence contains 245 $\times$ 64-bit block $\times$ 64-bit block = 1,003,520-bit output. The process is repeated 1,000 times with this setup.

### 3.1.3 Plaintext/Ciphertext Correlation

Plaintext/Ciphertext Correlation (PCC) examines the correlation between plaintext/ciphertext pairs and is generated using the ECB mode of operation. Every sample utilizes $Y$ blocks of random $y$-bit plaintext and a random $x$-bit. Every plaintext block is encrypted with the random $x$-bit key.

To obtain a minimum of 1,000,000-bit ciphertext, this process is to be repeated 15,625 times for RECTANGLE-80 or RECTANGLE-128. Each sequence contains 15,625 $\times$ 64-bit block = 1,000,000-bit output. Every derived block is the result of the XOR operation between the plaintext and its corresponding ciphertext block that is generated using ECB mode as shown in Figure 3. This procedure will be repeated 1,000 times for the 15,625 plaintext blocks using a random 80-bit key (RECTANGLE-80) or 128-bit key (RECTANGLE-128).

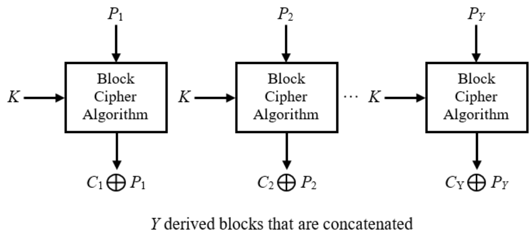


**Figure 3:** Plaintext/Ciphertext Correlation data category

#### 3.1.4 Cipher Block Chaining Mode

Ciphertext Block Chaining Mode (CBCM) is generated using the CBC mode of operation. Every plaintext block is XORed with the previous ciphertext block before being encrypted, whereas the initial block is XORed with an initialization vector as shown in Figure 4. Changes in the plaintext or initialization vector bits may affect the subsequent ciphertext blocks. Every sample utilizes all-zeros plaintext ($P$), a random $x$-bit key ($K$), and an all-zeros initialization vector ($IV$). The encryption process is applied for $I$ times.

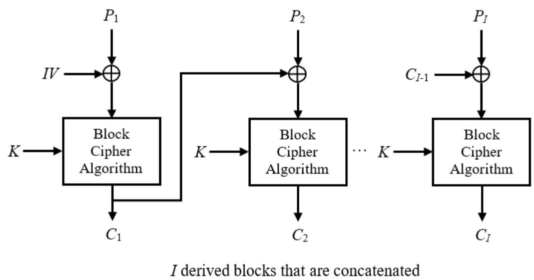


**Figure 4:** Cipher Block Chaining Mode data category

For CBCM, each sample utilizes 80 (RECTANGLE-80) or 128 (RECTANGLE-128) bits of random key, 15,625 blocks of 64-bit random plaintext, and a 64-bit all-zeros initialization vector ($IV$). The initial ciphertext block, $C_1$ is defined as $C_1 = (IV \oplus P_1)$, whereas the consecutive ciphertext blocks are defined as $C_i = E_k(C_{i-1} \oplus P_i)$ for $1 \leq i \leq 15,625$.

The derived blocks will be concatenated to obtain a large bit sequence. In order to obtain at least 1,000,000-bit output, every sequence will have 15,625 $\times$ 64-bit block = 1,000,000-bit output. This process is repeated until 1,000 sequences are generated.

### 3.1.5 Random Plaintext/Random Key

Random Plaintext/Random Key (RPRK) examines the randomness of ciphertext based on random plaintext and random key. Every sample utilizes $Y$ blocks of random $y$-bit plaintext and a random $x$-bit key. The plaintext block is encrypted with the random $x$-bit key as shown in Figure 5. The derived ciphertext blocks are generated using ECB mode of operation that will be concatenated to obtain a large bit sequence.

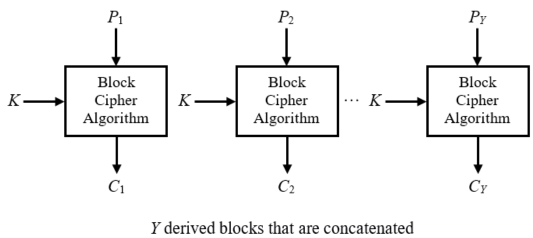


**Figure 5:** Random Plaintext/Random Key data category

To obtain a minimum of 1,000,000-bit ciphertext, the process is to be repeated 15,625 times for RECTANGLE-80 or RECTANGLE-128. Each sequence contains 15,625 $\times$ 64-bit block = 1,000,000-bit output. This procedure will be repeated 1,000 times for the 15,625 plaintext blocks using a random 80-bit key (RECTANGLE-80) or 128-bit key (RECTANGLE-128).

### 3.1.6 Low-Density Key

Low-Density Key (LDK) is constructed based on low-density $x$-bit keys. Every sample utilizes $Y$ blocks of random $y$-bit plaintext and $X$ blocks of a specific $x$-bit key. The initial 64-bit plaintext block is encrypted with an all-zeros

80-bit key (RECTANGLE-80) or 128-bit key (RECTANGLE-128). Next, the plaintext block is encrypted with the key with a single 1 in every bit position and other bits are set to 0s as shown in Figure 6. After that, the plaintext block is encrypted using a key with two 1s in every combination of two bits positions and other bits are set to 0s.

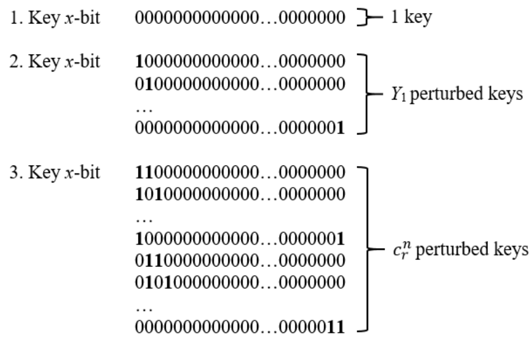


**Figure 6:** Low-Density Key data category

In total, the derived blocks for RECTANGLE-80 are $1 + C_1^{80} + C_2^{80} = 3,241$, while the derived blocks for RECTANGLE-128 are $1 + C_1^{128} + C_2^{128} = 8,257$. The ciphertext blocks are generated using ECB mode of operation that will be concatenated to obtain a large bit sequence. Each sequence contains 3,241 $\times$ 64-bit block = 207,424-bit output or 8,257 $\times$ 64-bit block = 528,448-bit output for RECTANGLE-80 and RECTANGLE-128 respectively. The procedure is then repeated to generate 1,000 sequences.

### 3.1.7 High-Density Key

High-Density Key (HDK) is constructed based on high-density $x$-bit. Every sample utilizes $Y$ blocks of random $y$-bit plaintext and $X$ blocks of a specific $x$-bit key. The initial 64-bit plaintext block is encrypted with an 80-bit key (RECTANGLE-80) or 128-bit key (RECTANGLE-128) of all 1s. Next, the plaintext block is encrypted using the key with a single 0 in every bit position and other bits are set to 1s as shown in Figure 7. After that, the plaintext block is encrypted using a key with two 0s in every combination of two bits positions and other bits are set to 1s.

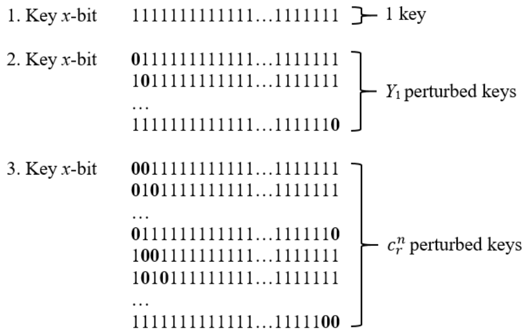


**Figure 7:** High-Density Key data category

In total, the derived blocks for RECTANGLE-80 are $1 + C_1^{80} + C_2^{80} = 3,241$, while the derived blocks for RECTANGLE-128 are $1 + C_1^{128} + C_2^{128} = 8,257$. The ciphertext blocks are generated using ECB mode of operation that will be concatenated to obtain a large bit sequence. Each sequence contains 3,241 $\times$ 64-bit block = 207,424-bit output or 8,257 $\times$ 64-bit block = 528,448-bit output for RECTANGLE-80 and RECTANGLE-128 respectively. The procedure is then repeated to generate 1,000 sequences.

### 3.1.8 Low-Density Plaintext

Low-Density Plaintext (LDP) is constructed based on low-density $y$-bit plaintext. Every sample utilizes $X$ blocks of random $x$-bit keys and $Y$ blocks of specific $y$-bit plaintext. First, the 64-bit plaintext block of all 0s is encrypted with an 80-bit key (RECTANGLE-80) or 128-bit key (RECTANGLE-128). Next, plaintext blocks with a single '1' in every bit position and other bits are set to '0' are encrypted using other random keys as shown in Figure 8. After that, the plaintext blocks with two 1s in every combination of two-bit position and other bits are set to 0 are encrypted using other random keys.

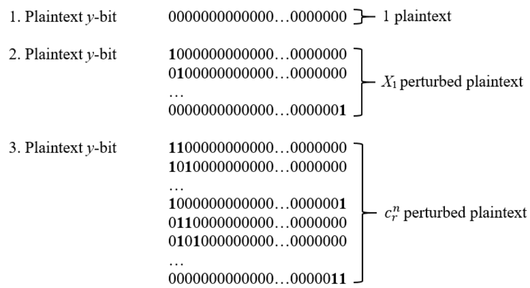


**Figure 8:** Low-Density Plaintext data category

In total, the derived blocks for RECTANGLE-80 and RECTANGLE-128 are $1 + C_1^{64} + C_2^{64} = 2,081$. The ciphertext blocks are generated using ECB mode of operation that will be concatenated to obtain a large bit sequence. Each sequence contains 2,081 $\times$ 64-bit block = 133,184 output. The procedure is then repeated to generate 1,000 sequences.

#### 3.1.9 High-Density Plaintext

High-Density Plaintext (HDP) is constructed based on high-density $y$-bit plaintext. Every sample utilizes $X$ blocks of random $x$-bit keys and $Y$ blocks of specific $y$-bit plaintext. First, the 64-bit plaintext block of all 1s is encrypted with an 80-bit key (RECTANGLE-80) or 128-bit key (RECTANGLE-128). Next, plaintext blocks with a single '0' in every bit position and other bits are set to '1' are encrypted using other random keys as shown in Figure 9. After that, the plaintext blocks with two 0s in every combination of two-bit position and other bits are set to 1 are encrypted using other random keys.

In total, the derived blocks for RECTANGLE-80 and RECTANGLE-128 are $1 + C_1^{64} + C_2^{64} = 2,081$. The ciphertext blocks are generated using ECB mode of operation that will be concatenated to obtain a large bit sequence. Each sequence contains 2,081 $\times$ 64-bit block = 133,184 output. The procedure is then repeated to generate 1,000 sequences.

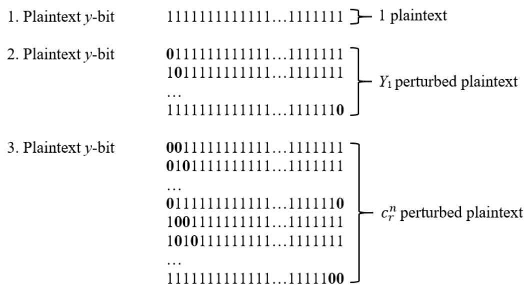


**Figure 9:** High-Density Plaintext data category

# 4 RESULTS AND ANALYSIS

Full 25 encryption rounds of RECTANGLE algorithm were performed to execute the randomness analysis. The tests were executed on RECTANGLE-80 and RECTANGLE-128. The random input data were generated using the random function from Microsoft Visual Studio 2008. Table 3 summarizes the required input bits recommended by the NIST (Rukhin et al. (2001)).

Several data categories produced by RECTANGLE block cipher are not able to complete all tests due to requirement limitations. Referring to Table 2 and Table 3, only SKA, SPA, PCC, CBCM, and RPRK data categories could be examined by all 15 tests (Abdullah et al. (2014)). Meanwhile, only ten tests can be executed with LDP and HDP. On the other hand, LDK and HDK can be executed by ten tests for RECTANGLE-80 and 11 tests for RECTANGLE-128. LDK, HDK, LDP, and HDP do not generate sufficient data as recommended by the NIST to execute the remaining tests.

| Statistical Test | Required No. of Bits |
|---|---|
| Runs | $n \geq 100$ |
| Frequency | |
| Block Frequency | |
| Cumulative Sums | |
| Longest Runs of Runs | $n \geq 128$ |
| Spectral DFT | $n \geq 1{,}000$ |
| Binary Matrix Rank | $n \geq 38{,}912$ |
| Maurer's Universal | $n \geq 387{,}480$ |
| Linear Complexity | $n \geq 1{,}000{,}000$ |
| Random Excursion | |
| Overlapping Templates | |
| Random Excursion Variant | |
| Serial | Not specified |
| Approximate Entropy | |
| Non-Overlapping Templates | |

**Table 3:** Required bits for each test.

The acceptable rejection range decides whether a sample passed or failed a statistical test. A sample passed a test if the rejected sequences fall within the specified range. Otherwise, if the rejected sequences fall outside of the range, the test failed. For Random Excursion and Random Excursion Variant tests, the evaluated samples are less than 1,000 due to an insufficient number of cycles as shown in Table 4. The N/A indicator shows that the test cannot be executed due to an insufficient sample.

| Statistical Test | No. of $p$-value(s) | | No. of Samples Evaluated | | Range of Acceptable Rejection | |
|---|---|---|---|---|---|---|
| | REC-80 | REC-128 | REC-80 | REC-128 | REC-80 | REC-128 |
| Runs | 1 | | 1,000 | | [0, 4] | |
| Frequency | | | | | | |
| Spectral DFT | | | | | | |
| Block Frequency | | | | | | |
| Linear Complexity | | | | | | |
| Maurer's Universal | | | | | | |
| Binary Matrix Rank | | | | | | |
| Approximate Entropy | | | | | | |
| Longest Runs of Ones | | | | | | |
| Overlapping Templates | | | | | | |
| Serial | 2 | | | | | |
| Cumulative Sums | | | | | | |
| Non-Overlapping Templates | 148 | | | | | |

| Statistical Test | | | Data Category | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | SKA | SPA | PCC | CBCM | RPRK | LDK | HDK | LDP | HDP |
| Random Excursion | No. of $p$-value(s) | REC-80 | 8 | | | | | | | | |
| | | REC-128 | | | | | | | | | |
| | No. of Samples Evaluated | REC-80 | 635 | 624 | 651 | 625 | 581 | N/A | | | |
| | | REC-128 | 645 | 627 | 628 | 622 | 648 | | | | |
| | Range of Acceptable Rejection | REC-80 | [0, 4] | [0, 3] | [0, 4] | [0, 3] | [0, 3] | | | | |
| | | REC-128 | [0, 4] | [0, 4] | [0, 4] | [0, 3] | [0, 4] | | | | |
| Random Excursion Variant | No. of $p$-value(s) | REC-80 | 18 | | | | | | | | |
| | | REC-128 | | | | | | | | | |
| | No. of Samples Evaluated | REC-80 | 635 | 624 | 651 | 625 | 581 | N/A | | | |
| | | REC-128 | 645 | 627 | 628 | 622 | 648 | | | | |
| | Range of Acceptable Rejection | REC-80 | [0, 4] | [0, 3] | [0, 4] | [0, 3] | [0, 3] | | | | |
| | | REC-128 | [0, 4] | [0, 4] | [0, 4] | [0, 3] | [0, 4] | | | | |

**Table 4:** Range of acceptable rejection for RECTANGLE-80 (REC-80) and RECTANGLE-128 (REC-128).

In general, RECTANGLE-80 passed 13 out of 15 statistical tests. The algorithm failed Random Excursion Variant and Non-Overlapping Templates tests. Meanwhile, RECTANGLE-128 passed Random Excursion, Random Excursion Variant, Runs, Linear Complexity, Binary Matrix Rank, Overlapping Templates, Block Frequency, Maurers Universal, Approximate Entropy, and Serial test. This RECTANGLE variant failed Non-Overlapping Templates, Frequency, Cumulative Sums, and Longest Runs of Ones tests.

The results as shown in Table 5 suggests that RECTANGLE does not pass all of the statistical tests. RECTANGLE-80 passed 1,556 out of 1,576 (98.73%) statistical tests, meanwhile RECTANGLE-128 passed 1,554 out of 1,578 (98.48%) statistical tests. In conclusion, RECTANGLE block cipher is not random based on the 0.1% significance level.

| | Runs | | Frequency | | Spectral DFT | | Block Frequency | | Linear Complexity | |
|---|---|---|---|---|---|---|---|---|---|---|
| Data Category | REC-80 | REC-128 | REC-80 | REC-128 | REC-80 | REC-128 | REC-80 | REC-128 | REC-80 | REC-128 |
| SKA | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| SPA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CBCM | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| RPRK | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| LDK | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | N/A | |
| 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 | | |
| | Maurer's Universal | | Binary Matrix Rank | | Approximate Entropy | | Longest Runs of Ones | | Overlapping Templates | |
| Data Category | REC-80 | REC-128 | REC-80 | REC-128 | REC-80 | REC-128 | REC-80 | REC-128 | REC-80 | REC-128 |
| SKA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| SPA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CBCM | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| RPRK | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| LDK | N/A | 0 | 0 | 0 | 0 | 0 | 0 | 0 | N/A | |
| HDK | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | |
| LDP | | | 0 | 0 | 0 | 0 | 0 | 0 | | |
| HDP | | | 0 | 0 | 0 | 0 | 0 | 0 | | |
| | Serial | | Cumulative Sums | | Non-Overlapping Templates | | Random Excursion | | Random Excursion Variant | |
| Data Category | REC-80 | REC-128 | REC-80 | REC-128 | REC-80 | REC-128 | REC-80 | REC-128 | REC-80 | REC-128 |
| SKA | 0 | 0 | 0 | 2 | 6 | 9 | 0 | 0 | 0 | 0 |
| SPA | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| PCC | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| CBCM | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| RPRK | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| LDK | 0 | 0 | 0 | 0 | 0 | 3 | N/A | | | |
| HDK | 0 | 0 | 0 | 0 | 2 | 1 | | | | |
| LDP | 0 | 0 | 0 | 0 | 3 | 2 | | | | |
| HDP | 0 | 0 | 0 | 0 | 5 | 3 | | | | |

**Table 5:** Number of rejected p-values for RECTANGLE-80 (REC-80) and RECTANGLE-128 (REC-128).

In particular, the algorithm failed most statistical tests in the SKA with six (RECTANGLE-80) and 13 (RECTANGLE-128) fails. The SKA data category is influenced by the sensitivity of an algorithm towards modifications of the cipher key. The finding reveals that the weakness of the key schedule algorithm is a factor that led to the randomness performance of RECTANGLE. These results proved that the RECTANGLE key schedule algorithm needs to be improved (Yan et al. (2019)).

The other finding that can be pointed out is RECTANGLE block cipher failed most of the Non-Overlapping Templates test with 19 (RECTANGLE-80) and 10 (RECTANGLE-128). The result indicates that RECTANGLE algorithm produces multiple output occurrences of a given non-periodic pattern. Therefore, it is necessary to enhance the RECTANGLE encryption algorithm.

# 5 CONCLUSION

An important principle in designing a block cipher is its ability to function as a pseudorandom number generator. The NIST Statistical Test Suite capable of evaluating the randomness criteria of a block cipher. The randomness of RECTANGLE has been analyzed using 1,000 samples and the results indicate that the block cipher is not random based on the 0.1% significance level. A cryptographic algorithm that passed all of the randomness tests does not guarantee its security strength (Isa and Z'aba (2014)). However, a secure cryptographic algorithm should pass all of the randomness tests. In the future, modifications on RECTANGLE block cipher are suggested to enhance its security.

# ACKNOWLEDGMENTS

The authors would like to extend our thanks to Universiti Sains Islam Malaysia for the resources funded and CyberSecurity Malaysia for the facilities provided. This research paper is supported by the Fundamental Research Grants No: FRGS/1/2019/ICT03/USIM/02/1.

# REFERENCES

Khan, M. A., and Salah, K. (2018). IoT security: review, blockchain solutions, and open challenges. *Future Generation Computer Systems*, 82: 395-411.

Öğünç, M. (2018). Differential cryptanalysis on LBLOCK using differential factors. (Master's thesis).

Poschmann, A. Y. (2009). Lightweight cryptography: cryptographic engineering for a pervasive world. (Ph.D thesis).

Guo, J., Peyrin, T., Poschmann, A., and Robshaw, M. (2011). The LED block cipher. *In International Workshop on Cryptographic Hardware and Embedded Systems*, 326-341. Springer, Berlin, Heidelberg.

Shibutani, K., Isobe, T., Hiwatari, H., Mitsuda, A., Akishita, T., and Shirai, T. (2011). Piccolo: An ultra-lightweight blockcipher. *In International Workshop on Cryptographic Hardware and Embedded Systems*, 342-357. Springer, Berlin, Heidelberg.

Tomoyasu, S. (2012). Twine: A lightweight block cipher for multiple platforms. *In Selected Areas in Cryptography*, 7707. Springer Berlin Heidelberg.

Zhang, W., Bao, Z., Lin, D., Rijmen, V., Yang, B., and Verbauwhede, I. (2015). RECTANGLE: A bit-slice lightweight block cipher suitable for multiple platforms. *Science China Information Sciences*, 58(12), 1-15.

Dinu, D., Perrin, L., Udovenko, A., Velichkov, V., Groschdl, J., and Biryukov, A. (2016). Design strategies for ARX with provable bounds: SPARX and LAX (full version). *IACR Cryptology ePrint Archive*, 2016, 984.

Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., and Wingers, L. (2015). The SIMON and SPECK lightweight block ciphers. *In Proceedings of the 52nd Annual Design Automation Conference*, 1-6.

Ariffin, S., and Yusof, N. A. M. (2017). Randomness analysis on 3D-AES block cipher. *In 2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery*, 331-335. IEEE.

Isa, H., and Z'aba, M. R. (2012). Randomness analysis on LED block ciphers. *In Proceedings of the Fifth International Conference on Security of Information and Networks*, 60-66.

Senol, A. (2017). Improved differential attacks on RECTANGLE. (Master's thesis).

Bao, Z., Luo, P., and Lin, D. (2015). Bitsliced implementations of the PRINCE, LED and RECTANGLE block ciphers on AVR 8-bit microcontrollers. *In International Conference on Information and Communications Security*, 18-36. Springer, Cham.

Omrani, T., Rhouma, R., and Sliman, L. (2018). Lightweight cryptography for resource-constrained devices: a comparative study and RECTANGLE cryptanalysis. *In International Conference on Digital Economy*, 107-118. Springer, Cham.

Zhang, W., Bao, Z., Rijmen, V., and Liu, M. (2015). A New classification of 4-bit optimal s-boxes and its application to PRESENT, RECTANGLE and SPONGENT. *In International Workshop on Fast Software Encryption*, 494-515. Springer, Berlin, Heidelberg.

Yan, H., Luo, Y., Chen, M., and Lai, X. (2019). New observation on the key schedule of RECTANGLE. *Science China Information Sciences*, 62(3): 32108.

Chew, L. C. N., Shah, I. N. M., Abdullah, N. A. N, Zawawi, N. H. A., Rani, H. A., and Zakaria, A. A. (2015). Randomness analysis on Speck family of lightweight block cipher. *International Journal of Cryptology Research*, 5(1): 44-60.

Aljohani, M., Ahmad, I., Basheri, M., and Alassafi, M. O. (2019). Performance analysis of cryptographic pseudorandom number generators. *IEEE Access*, 7: 39794-39805.

Tezcan, C., Okan, G. O., enol, A., Doan, E., Yceba, F., and Baykal, N. (2016). Differential attacks on lightweight block ciphers PRESENT, PRIDE, and RECTANGLE revisited. *In International Workshop on Lightweight Cryptography for Security and Privacy*, 18-32. Springer, Cham.

Feizi, S., Nemati, A., Ahmadi, A., and Makki, V. A. D. (2015). A high-speed FPGA implementation of a bit-slice ultra-lightweight block cipher, RECTANGLE. *In 2015 5th International Conference on Computer and Knowledge Engineering*, 206-211. IEEE.

Rukhin, A., Soto, J., Nechvatal, J., Smid, M., and Barker, E. (2001). A statistical test suite for random and pseudorandom number generators for cryptographic applications. *Booz-allen and hamilton inc mclean va*.

Simion, E., and Burciu, P. (2019). A note on the correlations between NIST cryptographic statistical tests suite. *University Politehnica of Bucharest Scientific Bulletin-Series A-Applied Mathematics and Physics*, 81(1): 209-218.

Sýs, M., Ríha, Z., Matyás, V., Marton, K., and Suciu, A. (2015). On the interpretation of results from the NIST statistical test suite. *Romanian Journal of Information Science and Technology*, 18(1): 18-32.

Moussaoui, S., Zeghdoud, S., and Allailou, B. (2019). Implementation and statistical tests of a block cipher algorithm MISTY1. *Malaysian Journal of Computing and Applied Mathematics*, 2(2): 44-59.

Abdullah, N. A. N., Lot, Zawawi, N. H. A., and Rani, H. A. (2011). Analysis on lightweight block cipher, KTANTAN. *In 2011 7th International Conference on Information Assurance and Security*, 46-51. IEEE.

Abdullah, N. A. N., Chew, L. C. N., Zakaria, A. A., Seman, K., and Norwawi, N. M. (2015). The comparative study of randomness analysis between modified version of LBlock block cipher and its original design. *International Journal of Computer and Information Technology*, 4(6): 867-875.

Abdullah, N. A. N., Seman, K., and Norwawi, N. M. (2014). Statistical analysis on LBlock block cipher. *In International Conference on Mathematical Sciences and Statistics 2013*, 233-245. Springer, Singapore.

Isa, H., and Z'aba, M. R. (2014). Randomness of the PRINCE block cipher. *International Conference on Frontiers of Communications, Networks and Applications*.

Zakaria, Abdul Alif and Azni, AH and Ridzuan, Farida and Zakaria, Nur Hafiza and Daud, Maslina (2020). Randomness analysis on RECTANGLE block cipher. *Proc. Cryptol. Inf. Secur. Conf.*, 133-142.